

PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD


Publication number: KR20050084629 (A)


Also published as:

Publication date: 2005-06-26


 WO2004042560 (A2)

Inventor(s): SCHULZ KENNETH R [US]; RAPP JOHN W [US]; JACKSON LARRY [US]; JONES MARK [US]; CHERASARO TROY [US]


 WO2004042560 (A3)

 WO2004042574 (A2)

Applicant(s): LOCKHEED CORP [US]

 WO2004042574 (A3)

Classification:

 WO2004042569 (A2)

- international: G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78; G06F9/30; G06F9/38; G06F9/445; G06F9/46; G06F15/78; (IPC1-7): G06F9/38

[more >>](#)

- European: G06F9/3834

Application number: KR20057007752 20050430

Priority number(s): US20030683929 20031009; US20030683932 20031009; US20030684053 20031009; US20030684057 20031009; US20030684102 20031009; US20020422503P 20021031

Abstract not available for KR 20050084629 (A)

Abstract of corresponding document: **WO 2004042560 (A2)**

A peer-vector machine includes a host processor and a hardwired pipeline accelerator. The host processor executes a program, and, in response to the program, generates host data, and the pipeline accelerator generates pipeline data from the host data. Alternatively, the pipeline accelerator generates the pipeline data, and the host processor generates the host data from the pipeline data. Because the peer-vector machine includes both a processor and a pipeline accelerator, it can often process data more efficiently than a machine that includes only processors or only accelerators. For example, one can design the peer-vector machine so that the host processor performs decision-making and non-mathematically intensive operations and the accelerator performs non-decision-making and mathematically intensive operations. By shifting the mathematically intensive operations to the accelerator, the peer-vector machine often can, for a given clock frequency, process data at a speed that surpasses the speed at which a processor-only machine can process the data.

.....
Data supplied from the esp@cenet database — Worldwide

KOREAN PATENT ABSTRACTS

(11) Publication number: 1020050084629 A

(43) Date of publication of application: 26.08.2005

(21) Application number: 1020057007752

(22) Date of filing: 30.04.2005

(30) Priority: 2003 683929 US 09.10.2003

(51) Int. Cl: G06F 9/38 (2006.01);

(71) Applicant: LOCKHEED MARTIN CORPORATION

(72) Inventor: SCHULZ KENNETH R.

RAPP JOHN W.

JACKSON LARRY

JONES MARK

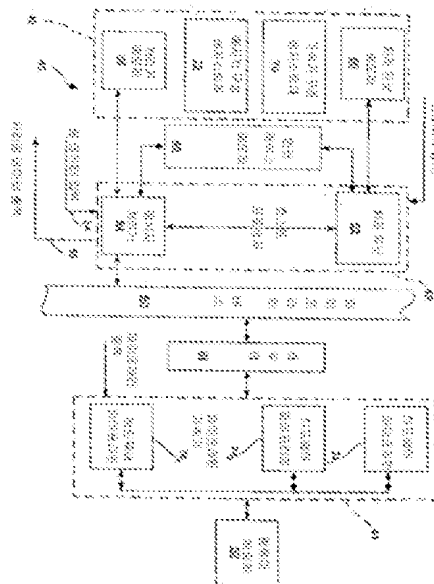
CHERASARO TROY

(54) PIPELINE ACCELERATOR HAVING MULTIPLE PIPELINE UNITS AND RELATED COMPUTING MACHINE AND METHOD

(57) Abstract:

A pipeline accelerator includes a bus and a plurality of pipeline units, each unit coupled to the bus and including at least one respective hardwired-pipeline circuit. By including a plurality of pipeline units in the pipeline accelerator, one can increase the accelerator's data-processing performance as compared to a single-pipeline-unit accelerator. Furthermore, by designing the pipeline units so that they communicate via a common bus, one can alter the number of pipeline units, and thus alter the configuration and functionality of the accelerator, by merely coupling or uncoupling pipeline units to or from the bus. This eliminates the need to design or redesign the pipeline-unit interfaces each time one alters one of the pipeline units or alters the number of pipeline units within the accelerator.

© KIPO & WIPO 2007



This Facsimile First Page has been artificially created from the Korean Patent Abstracts CD Rom

(19)대한민국특허청(KR)

(12) 공개특허공보(A)

(51) Int. Cl.⁷
G06F 9/38

(11) 공개번호 10-2005-0084629
(43) 공개일자 2005년08월26일

(21) 출원번호 10-2005-7007752
(22) 출원일자 2005년04월30일
 번역문 제출일자 2005년04월30일
(86) 국제출원번호 PCT/US2003/034555
 국제출원일자 2003년10월31일

(87) 국제공개번호 WO 2004/042561
 국제공개일자 2004년05월21일

(30) 우선권주장 10/683,929 2003년10월09일 미국(US)
10/683,932 2003년10월09일 미국(US)
10/684,053 2003년10월09일 미국(US)
10/684,057 2003년10월09일 미국(US)
10/684,102 2003년10월09일 미국(US)
60/422,503 2002년10월31일 미국(US)

(71) 출원인 특허드 마틴 코포레이션
미국 버지니아주 20110, 매나사스, 골든 드라이브 9500, 메일 드롭 043, 필딩 400

(72) 발명자 슐츠, 케네스 알.
미국 버지니아 20112, 매나사스, 골든 오를 코트 10506
렐, 존 더블류
미국 버지니아 20110, 매나사스, 리버 크레스트 로드 9350
잭슨, 테리
미국 버지니아 20112, 매나사스 크레스트북 드라이브 13093
존, 마크
미국 버지니아 20120, 셀트레빌, 오크메이 플레이스 15342
케라사르, 트로이
미국 버지니아 22701, 컬페퍼, 캐스트랄 코트 1524

(74) 대리인 전영일
영주석

심사청구 : 없음

(54) 다수의 파이프라인 유닛을 가지는 파이프라인 가속기 및관련 컴퓨팅 머신 및 방법

요약

파이프라인 가속기에는 버스 및 다수의 파이프라인 버스가 포함되어 있으며, 각 유닛은 상기 버스와 결합되어 있고 적어도 하나의 각각의 하드와이어드-파이프라인 회로를 포함한다. 파이프라인 가속기에 다수의 파이프라인 유닛을 포함시킴으로써, 설계자는 하나의 파이프라인-유닛 가속기와 비교해서 가속기의 데이터-처리 성능을 증가시킬 수 있다. 또한, 파이프라인 유닛이 통신 버스를 통해 통신하도록 파이프라인 유닛을 설계함으로써, 설계자는 파이프라인의 개수를 변경할

수 있어서 파이프라인 유닛을 상기 버스와 또는 버스로부터 결합하거나 결합을 해제하는 것만으로 가속기의 구성 및 기능을 변경할 수 있다. 이것은 가속기 내부의 파이프라인 개수를 변경하거나 또는 파이프라인 중 어느 하나를 변경하는 때마다 파이프라인-유닛 인터페이스를 설계하거나 재설계하는 수고를 없애준다.

대표도

도 3

색인어

파이프라인, 파이프라인 가속기, 하드와이어드-파이프라인

명세서

기술분야

(우선권 주장)

본 출원은 참고문헌으로서 통합되는 2002년 10월 31일 출원된 미국 가출원 제60/422,503호의 우선권을 주장한다.

(관련된 출원과의 상호참조)

본 출원은 발명의 명칭이 "향상된 컴퓨팅 아키텍처 및 관련 시스템 및 방법" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 가지는 컴퓨팅 머신 및 관련 시스템 및 방법" 인 미국 특허출원 제 10/684,053호, 발명의 명칭이 "향상된 컴퓨팅 아키텍처를 위한 파이프라인 가속기 및 관련 시스템 및 방법" 인 미국 특허출원 제10/683,929호, 및 발명의 명칭이 "프로그램가능한 회로 및 관련 컴퓨팅 머신 및 방법" 인 미국 특허출원 제10/684,057호와 관련이 되어 있으며, 상기 미국 특허출원 발명은 모두 2003년 10월 9일 동일한 권리자에 의해 출원되었으며, 본 명세서에 참고문헌으로 통합된다.

배경기술

상대적으로 짧은 시간에서 상대적으로 대용량의 데이터를 처리하기 위한 일반적인 컴퓨팅 아키텍처(computing architecture)에는 부하를 분배하는 다중 상호접속 프로세서(multiple interconnected processor)가 포함되어 있다. 처리 부하를 분배함으로써, 이들 다중 프로세서들은 주어진 클럭 주파수에서 하나의 프로세서가 할 수 있는 것 보다 더욱 빨리 데이터를 처리할 수 있다. 예를 들어, 프로세서 각각은 데이터의 일부를 처리 하거나 또는 처리되는 알고리즘 일부를 실행할 수 있다.

도 1은 다중-프로세서 아키텍처를 갖는 종래의 컴퓨팅 머신(computing machine)(10)의 개략적인 블록 다이어그램이다. 머신(10)에는 마스터 프로세서(12) 및 버스(16)를 통해 상기 마스터 프로세서와 상호 통신을 하는 코프로세서(14_1-14_n), 원격 장치(도 1에는 도시하지 않음)로부터 원 데이터(raw data)를 수신하는 입력 포트(18), 및 처리된 데이터를 상기 원격 소스로 제공하는 출력 포트(20)가 포함되어 있다. 상기 머신(10)에는 또한 마스터 프로세서(12)용 메모리(22), 코프로세서(14_1-14_n)용 메모리(24_1-24_n), 및 상기 버스(16)를 통해 마스터 프로세서와 코프로세서가 공유하는 메모리(26)도 포함되어 있다. 메모리(22)는 마스터 프로세서(12)를 위한 프로그램 및 작업 메모리 모두의 역할을 하고, 메모리(24_1-24_n) 각각은 코프로세서(14_1-14_n) 각각을 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 공유 메모리(26)는 마스터 프로세서(12)와 코프로세서(14)가 그들 사이의 데이터를 포트(18 및 20)를 통해 각각 원격 장치로/장치로부터 전달할 수 있게 해준다. 마스터 프로세서(12) 및 코프로세서(14)는 또한 머신(10)이 원 데이터를 처리하는 속도를 제어하는 공통의 클럭 신호를 수신하기도 한다.

일반적으로, 컴퓨팅 머신(10)은 마스터 프로세서(12)와 코프로세서(14) 간의 원 데이터 처리를 효과적으로 분배한다. 소나 어레이(sonar array)와 같은 원격 소스(도 1에는 도시하지 않음)는 포트(18)를 통해 원 데이터를 상기 원 데이터를 위한 선입선출(FIFO) 버퍼(도시하지 않음)로서 작동하는 공유 메모리(26)의 일부에 로드한다. 마스터 프로세서(12)는 버스(16)를 통해 메모리(16)로부터 원 데이터를 검색하고, 마스터 프로세서와 코프로세서(14)가 상기 원 데이터를 처리하고,

버스(16)를 통해 필요한 만큼 그들사이에서 데이터를 전달한다. 마스터 프로세서(12)는 처리된 데이터를 공유 메모리(26)에 정의되어 있는 다른 FIFO 버퍼(도시하지 않음)에 로드하고, 원격 소스가 포트(20)를 통해 이 FIFO로부터 상기 처리된 데이터를 검색한다.

다른 연산의 예에서, 컴퓨팅 머신(10)은 원 데이터를 처리하는데 있어서 상기 원 데이터상에서 각각의 연산에 $n+1$ 을 순차적으로 수행하여 처리하는데, 이 연산은 푸스트 주리에 변환(FFT)과 같은 처리 알고리즘을 함께 구성한다. 보다 특별하게는, 머신(10)은 마스터 프로세서(12)와 코프로세서(14)로부터 데이터-처리 파이프라인을 형성한다. 주어진 출력 신호의 주파수를 위해, 그러한 파이프라인은 종종 머신(10)이 하나의 프로세서만 가지는 머신보다 더욱 빠르게 원 데이터를 처리할 수 있게 한다.

메모리(26)내의 원-데이터 FIFO(도시하지 않음)로부터 원 데이터를 검색한 후, 마스터 프로세서(12)는 삼각 함수와 같은 제1 연산을 상기 원 데이터상에 수행한다. 이 연산은 프로세서(12)가 메모리(26) 내부에 정의된 제1-결과 FIFO(도시하지 않음)내에 저장하는 첫번째 결과를 산출해 낸다. 일반적으로, 프로세서(12)는 메모리(12) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 프로세서(12)는 또한 메모리(22)를 작업 메모리로 사용하여 프로세서가 상기 제1 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

다음으로, 상기 메모리(26)내의 제1-결과 FIFO(도시하지 않음)로부터 첫번째 결과를 검색한 후, 코프로세서(14)는 로그 함수와 같은 두번째 연산을 상기 첫번째 결과상에 수행한다. 이 두번째 연산은 코프로세서(14)가 메모리(26) 내부에 정의된 제2-결과 FIFO(도시하지 않음)내에 저장하는 두번째 결과를 산출해 낸다. 일반적으로, 코프로세서(14)는 메모리(24) 내에 저장된 프로그램을 수행하며, 그 프로그램의 제어하에 상기 설명된 연산들을 수행한다. 코프로세서(14)는 또한 메모리(24)를 작업 메모리로 사용하여 코프로세서가 상기 제2 연산의 중간 시간에 발생하는 데이터를 일시적으로 저장하기도 한다.

그리고 나서, 코프로세서(24₂-24_n)는 상기 두번째-($n-1$)번째 상에 세번째-n번째 연산을 순차적으로 수행하여 상기 코프로세서(24₁)를 위해 상기 설명한 것과 유사한 방법의 결과를 가져온다.

코프로세서(24_n)에 의해 수행되는 n 번째 연산은 최종 결과, 즉 처리된 데이터를 산출한다. 코프로세서(24_n)는 메모리(26) 내부에 정의된 처리된-데이터 FIFO(도시하지 않음)로 이 처리된 데이터를 로드(load)하고, 원격 장치(도 1에는 도시하지 않음)가 이 FIFO로부터 상기 처리된 데이터를 검색한다.

마스터 프로세서(12)와 코프로세서(14)가 처리 알고리즘의 다른 연산을 동시에 수행하기 때문에, 컴퓨팅 머신(10)은 서로 다른 연산을 순차적으로 수행하는 하나의 프로세서를 갖는 컴퓨팅 머신 보다도 원 데이터를 보다 빨리 처리할 수 있기도 하다. 특히, 하나의 프로세서는 원 데이터의 앞 세트상에 모든 $n+1$ 연산을 수행할 때 까지는 원 데이터의 새로운 세트를 검색할 수 없다. 그러나, 상기 언급한 파이프라인 기술을 이용해서, 마스터 프로세서(12)는 오직 첫번째 연산을 수행한 후 원 데이터의 새로운 세트를 검색할 수 있다. 따라서, 주어진 출력 주파수를 위해, 이 파이프라인 기술은 머신(10)이 원 데이터를 처리하는데 있어서 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 $n+1$ 배 더 빠른 원 데이터를 처리하는 속도를 증가시킬 수 있다.

선택적으로, 컴퓨팅 머신(10)은 원 데이터상에, FFT와 같은 처리 알고리즘의 경우에 $n+1$ 을 동시에 수행함으로써 병렬로 원 데이터를 처리할 수도 있다. 즉, 알고리즘에 있어서의 실시예에서 상기 언급한 바와 같은 $n+1$ 순차적 연산이 포함되어 있다면, 마스터 프로세서(12)와 코프로세서(14) 각각은 모든 $n+1$ 연산을 원 데이터 각각의 세트상에서 수행한다. 따라서, 주어진 출력 주파수를 위해서는, 상기 설명한 파이프라인 기술과 같이, 이러한 병렬-처리 기술은 머신(10)이 하나의 프로세서 머신(도 1에는 도시하지 않음)과 비교할 때 대략 $n+1$ 배 더 빠른 원 데이터 처리 속도를 증가시킬 수 있다.

불행하게도, 비록 컴퓨팅 머신(10)이 하나의 프로세서 컴퓨팅 머신(도 1에는 도시하지 않음)보다 빨리 데이터를 처리할 수는 있으나, 머신(10)의 데이터-처리 속도가 종종 프로세서 출력의 주파수보다 적어지곤 한다. 특히, 컴퓨팅 머신(10)의 데이터-처리 속도는 마스터 프로세서(12)와 코프로세서(14)가 데이터를 처리하는데 요구하는 시간에 의해 제한된다. 간략히 하기 위해, 이 속도 제한의 한 예를 마스터 프로세서(12)를 참고하여 설명하는데, 이 설명은 코프로세서(14)에도 적용됨을 이해할 수 있을 것이다. 앞에서 설명한 바와 같이, 마스터 프로세서(12)는 프로세서를 제어하여 데이터를 원하는 방식으로 조작하도록 제어하는 프로그램을 실행한다. 이 프로그램에는 프로세서(12)가 실행하는 일련의 명령이 포함되어 있다. 불행하게도, 프로세서(12)는 일반적으로 하나의 명령을 수행하는데 다중 출력 사이클을 필요로 하는데, 종종 다중 명령을 수행하여 데이터의 하나의 값을 처리해야 한다. 예를 들어, 프로세서(12)가 제1 데이터 값(A)(도시하지 않음)과 제2

데이터 값(B)(도시하지 않음)을 곱하는 경우를 가정한다. 제1 클럭 사이클 동안, 프로세서(12)는 메모리(22)로부터 여러개의 명령을 검색한다. 제2 및 제3 클럭 사이클 동안, 프로세서(12)는 메모리(22)로부터 A와 B를 각각 검색한다. 제4 클럭 사이클 동안, 프로세서(12)는 A와 B를 곱하고, 제5 클럭 사이클 동안, 그 결과물을 메모리(22 또는 26)에 저장하거나 또는 그 결과물을 원격 장치(도시하지 않음)로 제공한다. 이것은 최선의 시나리오인데, 그 이유는 많은 경우에서, 프로세서(12)는 카운터를 초기화(initializing) 및 닫는(closing) 경우와 같이 오버헤드 태스크(overhead task)를 위한 추가의 클럭 사이클을 요구하기 때문이다. 그러므로, 프로세서(12)는 A와 B를 처리하기 위해서는 5개의 클럭 사이클, 또는 데이터 값 당 평균 2.5개의 클럭 사이클을 필요로 한다.

따라서, 컴퓨팅 머신(10)이 데이터를 처리하는 속도는 종종 마스터 프로세서(12) 및 코프로세서(14)를 구동하는 클럭의 주파수보다 크게 낮아지곤 한다. 예를 들어, 프로세서(12)가 1.0 기가헤르츠(GHz)에서 클럭되지만 데이터 값 당 평균 2.5 클럭 사이클을 요구한다면, 유효 데이터-처리 속도는 $(1.0\text{GHz})/2.5=400\text{MHz}$ 가 될 것이다. 이 유효 데이터-처리 속도는 종종 초당 연산 단위로 측정되곤 한다. 그러므로, 1.0GHz 의 클럭 속도를 위해서는, 프로세서(12)는 초당 0.4 기가연산(Gops)의 데이터-처리 속도로 레이트(rate)되어야 한다.

도 2는 프로세서가 주어진 클럭 주파수 및 종종 파이프라인이 클럭되는 레이트의 거의 동일한 레이트에서 할 수 있는 것 보다 더 빠른 일반적인 데이터를 처리할 수 있는 하드와이어드(hardwired) 데이터 파이프라인(30)의 블록 다이어그램이다. 파이프라인(30)에는 각각 실행 프로그램 명령 없이 각각의 데이터상의 개별적 연산을 각각 수행하는 연산자 회로(32₁~32_n)가 포함되어 있다. 즉, 원하는 연산이 회로(32)로 "번 들어(burned in)" 것으로 프로그램 명령 없이도 자동적으로 연산을 실행하는 것이다. 실행 프로그램 명령과 관련된 오버헤드를 제어함으로써, 파이프라인(30)은 종종 주어진 클럭 주파수를 위해 할 수 있는 것 보다 더 많은 연산을 수행할 수 있다.

예를 들어, 파이프라인(30)은 프로세서가 주어진 클럭 주파수를 위해 할 수 있는 것 보다 더 빠른 다음과 같은 수식을 풀 수 있다.

$$Y(X_k)=(5X_k+3)2^{xk}$$

여기서, X_k 는 일련의 원 데이터 값을 나타낸다. 이 예에서, 연산자 회로(32₁)는 $5X_k$ 를 계산하는 곱셈기이고, 회로(32₂)는 $5X_k + 3$ 을 계산하는 가산기이며, 회로(32_n)($n=3$)는 $(5X_k + 3)2^{xk}$ 를 계산하는 곱셈기이다.

제1 클럭 사이클 $k=1$ 동안, 회로(32₁)는 데이터 값(X_1)을 수신하고 여기에 5를 곱해 $5X_1$ 을 발생한다.

제2 클럭 사이클 $k=2$ 동안, 회로(32₂)는 회로(32₁)로부터 $5X_1$ 을 수신하고, 3을 더해서 $5X_1 + 3$ 을 발생한다. 또한, 상기 제2 클럭 사이클 동안, 회로(32₁)는 $5X_2$ 를 발생한다.

제3 클럭 사이클 $k=3$ 동안, 회로(32₃)는 회로(32₂)로부터 $5X_1 + 3$ 을 수신하고, 2^{x1} ($x1$ 만큼 유효하게 $5X_1 + 3$ 왼쪽 시프트)를 곱하여 첫번째 결과($5X_1 + 3$) 2^{x1} 을 발생한다. 또한, 제3 클럭 사이클 동안, 회로(32₁)는 $5X_3$ 를 발생하고 회로(32₂)는 $5X_2 + 3$ 을 발생한다.

파이프라인(30)은 이러한 방식으로 모든 원 데이터 값이 처리될 때 까지 연속적인 원 데이터 값(X_k)을 계속 처리한다.

따라서, 원 데이터 값(X_1)을 수신한 후 두 개의 클럭 사이클의 지연 - 이 지연을 파이프라인(30)의 레이턴시(latency)라고 부르는 항 - 상기 파이프라인은 결과 $(5X_1 + 3)2^{x1}$ 을 발생하고, 그 후에 하나의 결과 - 예를 들어, $(5X_2 + 3)2^{x2}$, $(5X_3 + 3)2^{x3}$, ..., $(5X_n + 3)2^{xn}$, 를 각각의 클럭 사이클을 발생한다.

상기 레이턴시를 무시하면, 파이프라인(30)은 할록 속도와 동일한 데이터-처리 속도를 갖는다. 비교를 하면, 마스터 프로세서(12)와 코프로세서(14)가 상기 예에서와 같은 할록 속도의 0.4배의 데이터-처리 속도를 갖는다고 가정하면, 파이프라인(30)은 주어진 할록 속도를 위해 컴퓨팅 머신(10)(도 1)보다 2.5배 빨리 데이터를 처리할 수 있다.

도 2를 계속 참조하면, 설계자는 파이프라인(30)을 펠드-프로그래밍가능한 게이트 어레이(FPGA)와 같은 프로그래밍가능한 로직 IC(PLIC)에서 수행하도록 선택하기도 하는데, 그 이유는 PLIC는 주문형 반도체(ASIC)보다 나은 디자인 및 변형 유연성 가진다. PLIC 내부에서 하드와이어드 접속을 구성하기 위해서는, 설계자는 단지 PLIC 내부에 배치된 상호접속-구조 레지스터를 미리 결정된 이진 상태(binary state)로 설정하기만 하면 된다. 이들 이진 상태 모두의 조합을 "펌웨어(firmware)"라고 부르는 한다. 일반적으로, 설계자는 이 펌웨어를 PLIC 와 결합된 비휘발성 메모리(도 2에 도시하지 않음)에 로드한다. 누군가가 PLIC 를 "켜면(turn on)", PLIC는 상기 메모리로부터 펌웨어를 상기 상호접속-구조 레지스터로 다운로드한다. 그러므로, PLIC 의 기능을 변경시키기 위해서는, 설계자는 단지 펌웨어를 수정하면 되고 PLIC 로 하여금 그 수정된 펌웨어를 상호접속-구조 레지스터로 다운로드하게 하면 된다. 이것은 단지 펌웨어를 수정하는 것에 의해 PLIC를 수정할 수 있다는 것은 시계통화 단계 동안 및 "펠드 내에서" 파이프라인(30)의 업그레이드를 위해 특히 유용하다.

불행하게도, 하드와이어드 파이프라인(30)은 중요한 결정 형성, 특히 내포된 결정 형성(nested decision making)을 필요로 하는 알고리즘을 실행하는데 적선의 선택이 되지 못한다. 프로세서는 비교가능한 길이의 연산 명령(예를 들어, "A+B")을 실행할 수 있는 정도로 거의 빠르게 일반적으로 내포된-결정-형성 명령(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않으면 n" 과 같은 내포된 조건 명령)를 실행할 수 있다. 그러나, 비록 파이프라인(30)이 상대적으로 간단한 결정(예를 들어, "A>B?")을 유효하게 구성할 수 있어도, 일반적으로 내포된 결정(예를 들어, "A이면 B를 행하고, 그렇지 않고 C이면 D를 행하고, ... 그렇지 않고 n")을 프로세서가 할 수 있는 것 만큼 유효하게 실행할 수 없다. 이러한 비효율성의 한 이유는 파이프라인(30)은 온-보드 메모리(on-board memory)가 거의 없어서 외부 작업/프로그램 메모리(도시하지 않음)를 액세스 할 필요가 있기 때문이다. 그리고, 비록 그러한 내포된 결정을 실행하기 위해 파이프라인(30)을 디자인할 수 있다 하여도, 요구되는 회로의 크기와 복잡성은 종종 설계를 불가능하게 만드는데, 특히 여러개의 서로 다른 내포된 결정을 포함하는 알고리즘인 경우 그러하다.

따라서, 프로세서는 중요한 결정 형성을 요구하는 애플리케이션 내에서 종종 사용되며, 하드와이어드 파이프라인은 결정 형성이 거의 없는 또는 전혀 없는 "수치치리(number crunching)" 애플리케이션으로 제한되곤 한다.

더욱이, 아래에 설명한 바와 같이, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인, 특히 여러개의 PLIC를 포함하는 파이프라인(30)을 설계/수정하는 것 보다는 도 1의 컴퓨팅 머신(10)과 같은 프로세서-기반 컴퓨팅 머신을 설계/수정하는 것이 훨씬 쉽다.

프로세서와 그 주변장치들(예를 들어, 메모리)과 같은 컴퓨팅 구성요소들은 일반적으로 이 구성요소들의 상호접속을 촉진하여 프로세서-기반 컴퓨팅 머신을 형성하기 위한 산업-표준 통신 인터페이스를 포함한다.

특히, 표준 통신 인터페이스에는 두 개의 계층(layer)이 포함되는데, 물리 계층과 서비스 계층이다. 물리 계층에는 회로 및 이 회로의 연산 파라메터 및 인터페이스를 형성하는 대응 회로 상호접속이 포함되어 있다. 예를 들어, 물리 계층에는 구성요소를 버스와 연결하는 편, 이 편으로부터 데이터를 래치(latch)하는 버퍼, 및 이 편상에서 신호를 구동시키는 구동기가 포함되어 있다. 상기 연산 파라메터에는 상기 편이 수신하는 데이터 신호의 허용가능한 전압 범위, 데이터를 기록 및 판독하는 신호 타이밍, 및 지지된 연산 모드(예를 들어, 버스트 모드, 페이지 모드)가 포함되어 있다. 종래의 물리 계층에는 트랜지스터-트랜지스터 논리(TTL) 및 램버스(RAMBUS)가 포함된다.

서비스 계층에는 컴퓨팅 구성요소가 데이터를 전송하는 프로토콜이 포함된다. 이 프로토콜은 데이터의 포맷 및 상기 구성요소가 포맷된 데이터를 송수신하는 방식을 정의한다. 종래의 통신 프로토콜에는 파일-전송 프로토콜(FTP) 및 전송 제어 프로토콜/인터넷 프로토콜(TCP/IP)이 포함된다.

따라서, 산업-표준 통신 인터페이스를 갖는 제조자 및 다른 일반적인 설계 컴퓨팅 구성요소들로 인해서, 그러한 구성요소의 인터페이스를 일반적인 설계로 할 수 있고 이것을 상대적으로 적은 노력으로 다른 컴퓨팅 구성요소들과 상호접속시킬 수 있다. 이것은 설계자에게 대부분의 시간을 컴퓨팅 머신의 다른 부분들을 설계하는데 소비하게 만들고, 구성요소들을 추가하거나 없애는 것을 통해 머신을 쉽게 수정할 수 있게 한다.

산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 설계하는 것은 설계 라이브러리(design library)로부터 현존하는 물리-계층 설계를 사용하여 설계 시간을 절약하게 해 준다. 이것은 또한 구성요소들은 재고품인 컴퓨팅 구성요소들과 쉽게 접속할 수 있게 해주기도 한다.

공통의 산업-표준 통신 인터페이스를 지원하는 컴퓨팅 구성요소를 사용하여 컴퓨팅 머신을 설계하는 것은 설계자로 하여금 시간과 노력을 줄여주면서 구성요소들을 상호접속할 수 있게 해 준다. 이들 구성요소들이 공통의 인터페이스를 지원하기 때문에, 설계자는 설계 노력을 거의 들이지 않고 시스템 버스를 통해 이들을 상호 접속할 수 있다. 지원되는 인터페이스가 산업 표준이기 때문에, 설계자는 머신을 쉽게 수정할 수 있다. 예를 들어, 설계자는 다른 구성요소 및 주변장치들을 머신에 추가하여 시스템 설계를 발전시켜 나갈 수 있으며, 또는 차세대 구성요소들을 쉽게 추가/설계하여 기술 발전을 이룰 수 있다. 더욱이, 구성요소들이 공통의 산업-표준 서비스 계층을 지원하기 때문에, 설계자는 컴퓨팅 머신의 소프트웨어로 대응하는 프로토콜을 실현하는 현존하는 소프트웨어 모듈을 합체시킬 수 있다. 따라서, 설계자는 인터페이스 설계가 이미 적절하게 필수적이기 때문에 거의 노력을 들이지 않고 구성요소들을 접속시킬 수 있어서 머신이 특정 기능을 수행하게 하는 머신의 일부(예를 들어, 소프트웨어)를 설계하는 데 주력할 수 있다.

그러나, 불행하게도, 도 2의 파이프라인(30)과 같은 하드와이어드 파이프라인을 형성하는데 사용되는 PLIC 등과 같은 구성요소를 위한 알려진 산업-표준 서비스 계층은 없다.

따라서, 여러개의 PLIC 를 갖는 파이프라인을 설계하기 위해서, 설계자는 PLIC 간의 통신 인터페이스의 서비스 계층을 "스크래치(scratch)로부터" 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다. 일반적으로, ad hoc 서비스 계층은 PLIC 간에서 전달되는 데이터의 파라미터에 따라 달라진다. 비슷하게, 프로세서와 접속되는 파이프라인을 설계하기 위해서는, 설계자는 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

비슷하게, PLIC 을 추가하는 것으로 파이프라인을 수정하기 위해서는, 설계자는 일반적으로 추가된 PLIC와 현재의 PLIC 사이의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들이게 된다. 그리고, 프로세서를 추가하는 것으로 파이프라인을 수정하려면, 또는, 파이프라인을 추가하는 것으로 컴퓨팅 머신을 수정하려면, 설계자는 파이프라인과 프로세서간의 통신 인터페이스의 서비스 계층을 설계하고 디버깅하는데 상당한 시간과 노력을 들여야 한다.

그러므로, 도 1 및 도 2를 참고하면, 다수의 PLIC 를 접속하고 파이프라인과 프로세서의 접속 어려움으로 인해, 설계자는 컴퓨팅 머신을 설계하는 경우 상당한 트레이드오프(tradeoff)에 직면하곤 한다. 예를 들어, 프로세서-기반 컴퓨팅 머신을 가지고는, 설계자는 복잡한 결정-형성 가능성을 위한 트레이드 수-크런칭 속도 및 설계/수정 유연성에 집중하게 된다. 반대로, 하드와이어드 파이프라인-기반 컴퓨팅 머신을 가지고는, 설계자는 수-크런칭 속도를 위한 트레이드 복잡성-결정-형성 가능성 및 설계/수정에 집중하게 된다. 더욱이, 다수의 PLIC 를 접속하는 데의 어려움으로 인해, 소수의 PLIC 를 가지는 파이프라인-기반 머신을 설계하는 것이 불가능하기도 하다. 그 결과, 실제적인 파이프라인-기반 머신은 제한된 기능을 갖춘 한다. 그리고, 프로세서와 PLIC 와의 접속 어려움으로 인해서, 하나의 PLIC 이상과 프로세서와의 접속이 불가능하기도 하다. 그 결과, 프로세서와 파이프라인을 합체함으로써 얻어지는 이익이 적다.

따라서, 하드와이어드-파이프라인-기반 머신의 수-크런칭 속도를 가지고 프로세서-기반 머신의 결정-형성 가능성을 결합시킬 수 있는 새로운 컴퓨팅 아키텍처 요구가 있어 왔다.

발명의 상세한 설명

(요약)

본 발명의 한 실시예에 따르면, 파이프라인 가속기에는 버스 및 다수의 파이프라인 유닛이 포함되는데, 상기 파이프라인 유닛 각각은 상기 버스와 결합되어 있으며 적어도 하나의 각각의 하드와이어드-파이프라인 회로를 포함한다.

상기 파이프라인 가속기에 다수의 파이프라인 유닛을 포함시킴으로써, 가속기의 데이터-처리 성능을 하나의 파이프라인-유닛 가속기와 비교할 때 증가시킬 수 있다. 또한, 파이프라인 유닛이 공통의 버스를 통해 서로간 및 다른 피어들과 통신을 하도록 파이프라인 유닛을 설계함으로써, 설계자는 파이프라인 유닛의 수를 바꿀 수 있고, 그래서 단지 파이프라인 유

넷을 버스와 결합하거나 또는 결합을 해제하는 것 만으로 가속기의 구성과 기능을 바꾼다. 이것은 설계자가 가속기 내의 파이프라인 유닛의 하나를 바꾸거나 또는 파이프라인의 수를 바꾸는 경우마다 삼기 파이프라인-유닛 인터페이스를 설계하거나 재설계하는 수고를 없애준다.

도면의 간단한 설명

도 1은 종래의 다중-프로세서 아키텍처를 갖는 컴퓨팅 머신의 블록 다이어그램이고,

도 2는 종래의 하드와이어드 파이프라인의 블록 다이어그램이고,

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신의 블록 다이어그램이고,

도 4는 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기의 파이프라인 유닛의 블록 다이어그램이고,

도 5는 본 발명의 더 다른 실시예에 따른 도 3의 파이프라인 가속기의 파이프라인 유닛의 블록 다이어그램이고,

도 6은 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기의 블록 다이어그램이고,

도 7은 본 발명의 더 다른 실시예에 따른 도 3의 파이프라인 가속기의 블록 다이어그램이고,

도 8은 본 발명의 일 실시예에 따른 다중 파이프라인 유닛의 그룹을 포함하는 도 3의 파이프라인 가속기의 다이어그램이다.

실시예

(상세한 설명)

도 3은 본 발명의 일 실시예에 따른 피어-벡터 아키텍처를 갖는 컴퓨팅 머신(40)의 개략적인 블록 다이어그램이다. 호스트 프로세서(42)에 추가하여, 상기 피어-벡터 머신(40)에는 적어도 일부의 데이터 처리를 수행하여 도 1의 컴퓨팅 머신(10) 내의 코프로세서(14)의 뱅크(bank)를 유효하게 대체하는 파이프라인 가속기(44)가 포함되어 있다. 따라서, 호스트-프로세서(42) 및 가속기(44)(또는 후술할 유닛)는 데이터 벡터를 앞뒤로 전송할 수 있는 "피어(peer)"이다. 가속기(44)는 프로그램 명령을 실행하지 않으므로, 가속기는 종종 코프로세서의 뱅크가 주어진 클록 주파수에서 할 수 있는 것보다 훨씬 빠르게 데이터상의 집중적인 연산을 수학적으로 처리한다. 따라서, 프로세서(42)의 결정-형성 가능성과 가속기(44)의 수-크린성 가능성을 결합함으로써, 머신(40)은 동일한 능력을 갖지만, 머신(10)과 같은 종래의 컴퓨팅 머신보다는 빠르게 데이터를 처리할 수 있다. 또한, 후술하는 바와 같이, 가속기(44)에 상기 호스트 프로세서(42)의 통신 인터페이스와 호환되는 통신 인터페이스를 제공하는 것은 머신(40)의 설계 및 수정을 용이하게 해 주고, 특히, 프로세서의 통신 인터페이스가 산업 표준인 경우 그러하다. 가속기(44)에 다수의 파이프라인 유닛(예를 들어, PLIC-기반 회로)이 포함되어 있는 경우, 이들 유닛 각각에 동일한 인터페이스를 제공하는 것은 가속기의 설계 및 수정을 용이하게 해주는데, 특히 상기 통신 인터페이스가 산업-표준 인터페이스와 호환되는 경우 그러하다. 더욱이, 머신(40)은 후술하는 바와 같은 그리고 앞서 언급한 다른 출원들에서의 다른 장점도 제공한다.

도 3을 계속 참조하면, 호스트 프로세서(42) 및 파이프라인 가속기(44)에 추가하여, 피어-벡터 컴퓨팅 머신(40)에는 프로세서 메모리(46), 인터페이스 메모리(48), 파이프라인 버스(50), 하나 또는 그 이상의 펌웨어 메모리(52), 선택적인 원-데이터 입력 포트(54), 처리된-데이터 출력 포트(58), 및 선택적인 라우터(61)가 포함되어 있다.

호스트 프로세서(42)에는 처리 유닛(62) 및 메시지 처리기(64)가 포함되어 있으며, 프로세서 메모리(46)에는 처리-유닛 메모리(66) 및 처리기 메모리(68)를 포함하는데, 각각 프로세서 유닛 및 메시지 처리기를 위한 프로그램 및 작업 메모리 모두의 역할을 한다. 프로세서 메모리(46)에는 가속기-구성 레지스트리(70) 및 메시지-구성 레지스트리(72)도 포함되어 있는데, 이들은 각각 호스트 프로세서(42)로 하여금 가속기(44)의 기능 및 메시지 처리기(64)가 송수신하는 메시지의 포맷을 구성하도록 하는 각각의 구성 데이터를 저장하고 있다.

파이프라인 가속기(44)는 적어도 하나의 PLIC(도 4)에 배치되어 있으며 프로그램 명령을 실행하지 않고 각각의 데이터를 처리하는 하드와이어드 파이프라인(74₁-74_n)을 포함하고 있다. 펌웨어 메모리(52)는 가속기(44)용 구성 펌웨어를 저

강한다. 단일 가속기(44)가 다수의 PLIC 에 배치된다면, 이들 PLIC 및 그들 각각의 펌웨어 메모리는 다중 파이프라인 유닛에 배치되며, 도 4 내지 도 8을 참고로 아래에 설명되어 있다. 선택적으로, 상기 가속기(44)는 적어도 하나의 ASIC 에 배치될 수 있으며, 따라서, 일단 ASIC 가 형성되면, 구성할 수 없는 내부 상호접속을 갖는다. 이 대안에서, 머신(40)에서 펌웨어 메모리(52)를 생략할 수 있다. 또한, 비록 가속기(44)가 다중 파이프라인(74₁~74_n)을 포함하는 것으로 도시되어 있으나, 오직 하나의 파이프라인만 포함할 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 디지털-신호 처리기(DSP)와 같은 하나 또는 그 이상의 프로세서가 포함될 수 있다. 또한, 비록 도시하지는 않았지만, 가속기(44)에는 데이터 입력 포트 및/또는 데이터 출력 포트를 포함할 수 있다.

비록 호스트 프로세서(42) 및 파이프라인 가속기(44)를 서로 다른 IC에 배치되어 있는 것으로 설명하였으나, 호스트 프로세서와 파이프라인 가속기는 동일한 IC 상에 배치되어도 좋다.

파이-벡터 머신(40)의 일반적인 동작은 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에 설명되어 있고, 상기 호스트 프로세서(42)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,053호에 설명되어 있으며, 파이프라인 가속기(44)의 구조 및 동작은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에 설명되어 있으며, 도 4 내지 도 8을 참고하여 후술한다.

도 4는 본 발명의 일 실시예에 따른 도 3의 파이프라인 가속기(44)의 유닛(78) 블록 다이어그램이다.

가속기(44)에는 하나 또는 그 이상의 그러한 파이프라인 유닛(78)(도 4에는 하나만 도시함)이 포함되어 있는데, 각각의 유닛에는 PLIC 또는 ASIC 같은 파이프라인 회로(80)가 포함되어 있다. 후술하고 상기 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에 설명된 바와 같이, 각각의 파이프라인 유닛(78)은 호스트 프로세서(42)(도 3) 및 가속기(44)의 다른 파이프라인 유닛의 "피어"이다. 즉, 파이프라인 유닛(78) 각각은 호스트 프로세서(42) 또는 다른 파이프라인 유닛과 직접 통신할 수 있다. 따라서, 이 피어-벡터 아키텍처는 파이프라인 유닛(78) 모두가 마스터 파이프라인 유닛(도시하지 않음) 또는 호스트 프로세서(42)와 같은 중앙 위치를 통해서 통신하는 경우 발생하기도 하는 데이터의 "병목현상"을 막아준다. 또한, 이 구조는 설계자에게 머신을 크게 수정하지 않고 피어-벡터 머신(40)(도 3)으로부터 피어를 추가 또는 제거할 수 있게 해준다.

파이프라인 회로(80)에는 호스트 프로세서(42)(도 3)와 같은 피어와 다음과 같은 파이프라인 회로의 다른 구성요소들 사이에서 데이터를 전달해주는 통신 인터페이스(82)가 포함되어 있다. 상기 다른 구성요소들로는 통신 셀(84)을 통한 하드웨어도 파이프라인(74₁~74_n), 파이프라인 게이트(86), 예외 관리자(88), 및 구성 관리자(80)가 있다. 파이프라인 회로(80)에는 산업-표준 버스 인터페이스(91) 및 상기 인터페이스(82)와 상기 인터페이스(91)를 연결하는 통신 버스(93)도 포함되어 있다. 선택적으로, 인터페이스(91)의 기능은 통신 인터페이스(82) 내에 포함되어 있으며, 상기 버스(93)는 생략해도 좋다.

파이프라인 회로(80)의 구성요소들을 별개 모듈로 설계함으로써, 설계자는 파이프라인 회로의 설계를 간단하게 할 수 있다. 즉, 설계자는 이들 구성요소 각각을 개별적으로 설계 및 검사할 수 있으며, 소프트웨어 또는 프로세서-기반 컴퓨팅 시스템(도 1의 시스템(10)과 같은)을 설계하는 경우 이들을 하나로 집적할 수 있다. 또한, 설계자는 이들 구성요소, 특히 설계자가 다른 파이프라인 설계에서 자주 사용하는 통신 인터페이스(82)와 같은 구성요소를 정의하는 라이브러리 하드웨어 설명 언어(HDL)(도시하지 않음)를 세이브(save)할 수 있어서, 동일한 구성요소를 사용하는 미래의 파이프라인 설계를 검사하고 설계하는 시간을 줄여준다. 즉, 라이브러리에서 HDL 을 사용함으로써, 설계자는 이전에 구현된 구성요소들을 스크래치로부터 다시 설계할 필요가 없어지므로, 이전에는 구현되지 못한 구성요소의 설계 또는 이전에 구현된 구성요소들의 수정에 노력을 집중할 수 있다. 더욱이, 파이프라인 회로(80) 또는 가속기(44)의 전체 파이프라인의 여러 버전(version)을 정의하는 라이브러리 HDL 내에 세이브할 수 있어서, 현존하는 디자인 가운데에서 고르고 선택할 수 있다.

도 4를 계속 참고하면, 통신 인터페이스(82)는 메시지 처리기(64)(도 3)에 의해 확인된 포맷으로 데이터를 송수신(준제한다면 버스 인터페이스(91)를 통해서)해서, 일반적으로 피어-벡터 머신(40)(도 3)의 설계 및 수정을 용이하게 한다. 예를 들어, 단일 데이터 포맷이 Rapid I/O 포맷과 같은 산업 표준인 경우, 설계자는 호스트 프로세서(42)와 파이프라인 유닛(78) 사이의 맞춤형 접속을 설계할 필요는 없다. 또한, 논-버스 인터페이스를 통해서가 아니라 파이프라인 버스(50)를 통해서 호스트 프로세서(42)(도 3)와 같은 다른 피어와 파이프라인 유닛(78)이 통신하도록 하게 함으로써, 설계자는 단지 이

들을 파이프라인 유닛이 추가되거나 제거되는 시간마다 스크래치로부터 논-비스 인터페이스를 재설계하는 대신 파이프라인 비스와 어플(또는 어플을 보유하고 있는 회로 카드)을 접속하거나 접속해제하는 것만으로 파이프라인 유닛(78)의 수를 변경할 수 있다.

하드웨어 하드 파이프라인(74₁~74_n)은 도 3을 참고하여 앞서 설명하고, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호 및 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에서 설명된 개별적으로 데이터상의 연산을 수행하며, 통신 셀(84)은 상기 파이프라인과 상기 파이프라인(80)의 다른 구성요소들 및 상기 파이프라인 유닛(78)의 다른 회로(후술하는 데이터 메모리(92)와 같은) 접속시킨다.

제어기(86)는 SYNC 신호 및 다른 피어로부터의 특별한 파이프라인-비스 통신(즉, "이벤트")에 응답하여 하드웨어 하드 파이프라인(74₁~74_n)을 동기화 하고, 파이프라인이 그들 각각 데이터 연산을 수행하는 시퀀스를 모니터 및 제어한다. 예를 들어, 호스트 프로세서(42)와 같은 피어는 SYNC 신호를 펄스하거나 또는 파이프라인 비스(50)를 통해 파이프라인 유닛(78)으로 이벤트를 송신하여 피어가 상기 파이프라인 유닛으로 데이터의 블록 전송을 완료했음을 표시하고 하드웨어 하드 파이프라인(74₁~74_n)이 이 데이터 처리를 시작하게 한다.

일반적으로, SYNC 신호는 시간-임계 연산을 동기화 하고, 어느 이벤트는 비-시간-임계 연산을 동기화 하는 데 사용된다. 일반적으로, 어느 이벤트는 "도어벨(doorbell)"이라 불리는 데이터-없는 통신이다. 그러나, 어느 이벤트는 "이벤트 메시지"로 불리는 데이터를 포함하기도 한다. SYNC 신호 및 이벤트는 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

예외 관리자(88)는 상기 하드웨어 하드 파이프라인(74₁~74_n), 통신 인터페이스(82), 통신 셀(84), 제어기(86), 및 비스 인터페이스(91)(존재한다면)의 상태를 모니터하고, 호스트 프로세서(42)(도 3)에게 예외를 보고한다. 예를 들어, 통신 인터페이스(82)내의 버퍼가 오버플로우(overflow) 되면, 예외 관리자(88)는 이를 호스트 프로세서(42)에게 보고한다. 예외 관리자는 상기 예외로 발생하는 문제를 해결하거나 또는 해결을 시도하기도 한다. 예를 들어, 버퍼의 오버플로우를 위해서, 예외 관리자(88)는, 후술하는 바와 같이, 구성 관리자(90)를 통해 또는 직접 버퍼의 크기를 증가시킨다.

구성 관리자(90)는, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD"인 미국 특허출원 제10/684,102호에 설명된 바와 같이, 호스트 프로세서(42)(도 3)로부터 소프트웨어-구성 데이터에 응답하여 하드웨어 하드 파이프라인(74₁~74_n), 통신 인터페이스(82), 통신 셀(84), 제어기(86), 예외 관리자(88), 및 인터페이스(91)(존재한다면)의 소프트웨어 구성(software configuration)을 설정하고, 하드 구성은 파이프라인 회로(80)의, 트랜지스터 및 회로-블록 배열상의 실제 토폴로지를 표시하고, 소프트웨어 구성은 상기 하드-구성된 구성요소의 물리적 파라미터(예를 들어, 데이터 폭, 데이터 크기)를 표시한다. 즉, 소프트웨어 구성 데이터는 프로세서의 연산 모드(예를 들어, 비스-메모리 모드)를 설정하기 위해 프로세서의 레지스터(도 4에는 도시하지 않음)로 로드될 수 있는 데이터와 유사하다. 예를 들어, 호스트 프로세서(42)는 상기 구성 관리자(90)로 하여금 통신 인터페이스(82)내의 데이터 및 이벤트 큐의 수 및 각각의 우선 레벨을 설정하도록 만드는 소프트웨어-구성 데이터를 송신한다. 예외 관리자(88)는 상기 구성 관리자(90)로 하여금, 예를 들어 통신 인터페이스(82) 내의 버퍼의 오버플로우 크기를 증가시키도록 하는 소프트웨어-구성 데이터를 송신하기도 한다.

산업-표준 비스 인터페이스(91)는 통신 인터페이스로부터 인터페이스 회로의 일부를 효과적으로 오프로드함으로써 상기 통신 인터페이스(82)의 크기와 복잡성을 줄여주는 종래의 비스-인터페이스 회로이다. 따라서, 설계자가 파이프라인 비스(50) 또는 라우터(61)(도 3)의 파라미터를 바꾸고 싶으면, 설계자는 통신 인터페이스(82)가 아니라 인터페이스(91)를 수정하기만 하면 된다. 선택적으로, 설계자는 상기 인터페이스(91)를 상기 파이프라인 회로(80)의 외부에 있는 IC(도시하지 않음)내에 배치할 수 있다. 상기 인터페이스(91)를 파이프라인 회로(80)로부터 오프로드시키는 것은 하드웨어 하드 파이프라인(74₁~74_n) 및 제어기(86) 등에서의 사용을 위한 상기 파이프라인 회로상의 리소스를 자유롭게 해준다. 또는, 위에서 설명한 바와 같이, 비스 인터페이스(91)는 통신 인터페이스(82)의 일부일 수 있다.

도 4를 계속 참고하면, 파이프라인 회로(80)에 추가하여, 가속기(44)의 파이프라인 유닛(78)에는 데이터 메모리(92)가 포함되어 있고, 파이프라인 회로가 PLIC라면, 펌웨어 메모리(52)가 포함된다.

데이터 메모리(92)는 데이터가 호스트 프로세서(42)(도 3)와 같은 다른 피어와 하드와이어드 파이프라인(74₁~74_n) 사이에서 이동하도록 비퍼작용을 하고, 상기 하드와이어드 파이프라인을 위한 작업 메모리가 되기도 한다. 통신 인터페이스(82)는 데이터 메모리(82)와 파이프라인 버스(50)를 접속 시키고(통신 버스(94) 및 존재한다면 산업-표준 인터페이스(91)를 통해서), 통신 셀(84)은 데이터 메모리를 상기 하드와이어드 파이프라인(74₁~74_n)과 접속시킨다.

데이터 메모리(92)(또는 파이프라인 유닛(78)의 다른 부분)은 파이프라인 유닛의 프로파일을 저장할 수 있다. 이 프로파일은 상기 호스트 프로세서(42)(도 3)가 자기 자신을 적절히 구성하고, 내부간 통신을 위해 피어-팩터 머신(40)(도 3)의 다른 피어 및 파이프라인 유닛으로 상기 파이프라인 유닛(78)을 충분히 디스크라이브 한다. 예를 들어, 상기 프로파일은 상기 파이프라인 유닛(78)이 수행할 수 있는 데이터 연산 및 통신 프로토콜을 식별한다. 따라서, 피어-팩터 머신(40)의 초기화 동안 상기 프로파일을 관독함으로써, 호스트 프로세서(42)는 상기 메시지 처리기(64)(도 3)를 적절히 구성하여 파이프라인 유닛(78)과 통신할 수 있다. 이 기술은 컴퓨터 자신이 디스크 드라이브 등의 새롭게 설치된 주변장치와 통신할 수 있는 "플러그 앤드 플레이" 기술로 알려져 있다. 호스트 프로세서(42) 및 파이프라인 유닛(78)의 구성은 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,102호에 더 설명되어 있다.

도 3을 참고하여 위에서 설명한 바와 같이, 파이프라인 회로(80)는 PLIC 이고, 펌웨어 메모리(52)는 상기 파이프라인 회로의 하드 구성을 설정하는 상기 펌웨어를 저장한다. 메모리(52)는, 가속기(44)의 구성 동안에 상기 펌웨어를 파이프라인 회로(80)에 로드하고, 상기 가속기의 구성 동안에 또는 구성이 이루어진 후에 상기 통신 인터페이스(82)를 통해 호스트 프로세서(42)(도 3)으로부터 수정된 펌웨어를 수신한다. 펌웨어를 로드하고 수신하는 것은 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호 및 발명의 명칭이 "PROGRAMMABLE CIRCUIT AND RELATED COMPUTING MACHINE AND METHOD" 인 미국 특허출원 제10/684,057호에 더 설명되어 있다.

도 4를 계속 참고하면, 파이프라인 유닛(78)은 파이프라인 회로(80), 데이터 메모리(92) 및 펌웨어 메모리(52)가 배치되는 회로 보드 또는 카드(98)를 포함한다. 상기 회로 보드는 더터 카드(daughter card)가 개인용 컴퓨터(도시하지 않음)내의 머더보드의 슬롯으로 꽂힐 수 있는 것처럼 파이프라인-버스 커넥터(도시하지 않음)로 꽂힌다. 비록 도시하지는 않았으나, 상기 파이프라인 유닛(78)에는 종래의 IC 및 전력 조정기(power regulator)나 전력 시퀀서와 같은 구성요소도 포함하며, 이들 IC/구성요소들은 알려진 바와 같이 상기 카드(98)상에 배치되기도 한다.

상기 파이프라인 유닛(78)의 구조 및 동작의 보다 상세한 사항은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명되어 있다.

도 5는 본 발명의 더 다른 실시예에 따른 도 3의 파이프라인 가속기(44)의 파이프라인 유닛(100)의 블록 다이어그램이다. 파이프라인 유닛(100)은, 다수의 파이프라인 회로(80) - 여기에서는 두 개의 파이프라인 회로(80a, 80b) - 가 포함되어 있다는 것을 제외하고는 도 4의 파이프라인 유닛(78)과 유사하다. 파이프라인 회로(80)의 수를 늘이는 것은 하드와이어드 파이프라인(74₁~74_n)의 개수(n)를 증가시켜 파이프라인 유닛(78)과 비교할 때 파이프라인 유닛(100)의 기능을 증가시킨다. 또한, 상기 파이프라인 유닛(100)에는 파이프라인 회로(80a)를 위한 펌웨어 메모리(52a) 및 파이프라인 회로(80b)를 위한 펌웨어 메모리(52b)를 포함한다. 선택적으로, 파이프라인 회로(80a, 80b)는 하나의 펌웨어 메모리를 할당하기도 한다.

상기 파이프라인 유닛(100)에서, 서비스 구성요소, 즉, 통신 인터페이스(82), 제어기(86), 예외 관리자(88), 구성 관리자(90) 및 선택적인 산업-표준 버스 인터페이스(91)가 상기 파이프라인 회로(80a)상에 배치되며, 파이프라인(74₁~74_n) 및 통신 셀(84)은 파이프라인 회로(80b)상에 배치된다. 상기 서비스 구성요소와 파이프라인(74₁~74_n)을 별개의 파이프라인 회로(80a, 80b)에 각각 위치시킴으로써, 설계자는 상기 서비스 구성요소 및 파이프라인이 동일한 파이프라인 회로상에 위치시키고 싶은 것 보다 더 많은 파이프라인의 수(n) 및/또는 더 복잡한 파이프라인을 포함시킬 수 있다. 선택적으로, 파이프라인(74₁~74_n)을 인터페이스(82)로 인터페이스하고 제어기(86)로 인터페이스하는 통신셀(84)의 일부가 파이프라인 회로(80a)상에 배치될 수 있다.

상기 파이프라인 유닛(100)의 구조 및 동작에 대한 더 자세한 내용은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명되어 있다.

도 6은 본 발명의 일 실시예에 따른 다중 파이프라인 유닛(78)(도 4) 또는 (100)(도 5)을 갖는 도 3의 가속기(44)의 블록 다이어그램이다. 간략화를 위해, 가속기(44)를 여러 파이프라인 유닛($78_1 \sim 78_n$)을 가지는 것으로 설명하였으나, 상기 가속기에는 다중 파이프라인 유닛(100) 또는 유닛(78)과 유닛(100)의 결합이 포함될 수 있다. 다중 파이프라인 유닛(78)을 포함함으로써, 설계자는 하나의 파이프라인 유닛만 가지는 가속기와 비교할 때 가속기(44)의 기능 및 처리 능력을 증가시킬 수 있다. 또한, 각각의 파이프라인 유닛(78)은 일반적으로 공통의 산업-표준 인터페이스를 가지고 있기 때문에, 설계자는 파이프라인 유닛을 추가하거나 제거하는 것으로 가속기(44)를 쉽게 수정할 수 있다.

다중-파이프라인 가속기(44)의 한 수행에서, 산업-표준 버스 인터페이스(91)는 각각의 파이프라인 유닛($78_1 \sim 78_n$)에서 생략되고, 하나의 외부(파이프라인 유닛의 외부) 인터페이스(91) 및 통신 버스(94)는 모든 파이프라인 유닛에게 공통이다. 하나의 외부 버스 인터페이스(91)를 포함시키는 것은 도 4를 참고로 앞서 설명한 바와 같이 파이프라인 회로(80)(도 4)상의 리소스를 자유롭게 해준다. 상기 파이프라인 유닛($78_1 \sim 78_n$)은 하나의 회로 보드(도 6에는 도시하지 않음)상에 모두 배치될 수 있고, 각각의 파이프라인 유닛은 각각의 회로 보드상에 배치될 수 있고, 또는 다수의 파이프라인 유닛의 그룹이 다수의 회로 보드상에 각각 배치될 수 있다. 후자의 두 경우에서, 버스 인터페이스(91)가 하나의 회로 보드상에 배치된다. 선택적으로, 파이프라인 유닛($78_1 \sim 78_n$) 각각에는 도 4를 참고하여 앞서 설명한 바와 같이 각각의 산업-표준 버스 인터페이스(91)가 포함되어 있어서, 각각 상기 파이프라인 버스(50) 또는 라우터(61)(도 3)과 직접 통신할 수 있다. 이 수행에서, 파이프라인 유닛($78_1 \sim 78_n$)은 상기 설명한 바와 같이 하나 또는 다수의 회로 보드상에 배치될 수 있다.

파이프라인 유닛($78_1 \sim 78_n$) 각각은 상기 호스트 프로세서(42)(도 3)의 피어 및 서로 간의 피어이다. 즉, 각각의 파이프라인 유닛(78)은 통신 버스(94)를 통해 다른 파이프라인 유닛과 직접 통신할 수 있으며, 통신 버스(94), 버스 인터페이스(91), 라우터(61)(존재한다면), 및 파이프라인 버스(50)를 통해 호스트 프로세서(42)와 통신할 수 있다. 선택적으로, 파이프라인 유닛($78_1 \sim 78_n$) 각각에 개개의 버스 인터페이스(91)가 포함되어 있으면, 각각의 파이프라인 유닛은 라우터(61)(존재한다면) 및 파이프라인 버스(50)를 통해 호스트 프로세서(42)와 직접 통신할 수 있다.

상기 다중-파이프라인-유닛 가속기(44)의 동작을 두 개의 예를 통해 설명한다.

첫번째 실시예로서, 파이프라인 유닛(78_1)은 시간-임계 방식으로 상기 데이터를 처리하는 파이프라인(78_n)으로 데이터를 전송한다. 그래서, 파이프라인 유닛($78_1, 78_n$)은 하나 또는 그 이상의 SYNC 신호를 사용하여 상기 데이터 전송 및 처리를 동기화한다. 일반적으로, SYNC 신호는 시간-임계 함수를 트리거 하기에는 충분히 빠르지만, 특정한 하드웨어 리소스를 필요로 한다. 비교적, 이벤트는 시간-임계 기능을 트리거하기에는 충분히 빠르지만, 하드웨어 리소스를 거의 필요로 하지 않는다. 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 10/683,929호에 설명되어 있는 바와 같이, SYNC 신호가 피어에서 피어까지 직접 경로설정이 되기 때문에, 파이프라인 버스(50)(도 3) 및 통신 버스(94) 등을 주사(traverse)하는 이벤트보다 훨씬 빠르게 기능을 트리거한다. 그러나, 이들은 별도로 경로 설정이 되기 때문에, SYNC 신호는, 파이프라인 회로(80)(도 4)의 라우팅 라인 및 버퍼 π 의 전용 회로를 필요로 한다. 반대로, 이들은 현존하는 데이터-전송 기반시설(예를 들어, 파이프라인 버스(50) 및 통신 버스(94))를 사용하기 때문에, 상기 이벤트는 전용 하드웨어 리소스를 거의 요구하지 않는다. 따라서, 설계자는 이벤트를 사용하여 대부분의 시간-임계 기능을 트리거 한다.

먼저, 파이프라인 유닛(78_1)은 데이터를 통신 버스(94)상으로 구동시킴으로써 파이프라인 유닛(78_n)으로 데이터를 전송한다. 일반적으로, 파이프라인 유닛(78_1)은 데이터를 포함하는 메시지 및 상기 파이프라인 유닛(78_n)의 어드레스를 포함하는 헤더를 생성한다. 만약 파이프라인 유닛(78_1)이 상기 데이터를 다중 파이프라인 유닛(78)으로 전송한다면, 상기 두 방법 중 하나의 방법으로 하게 된다. 특히, 파이프라인 유닛(78_1)은 상기 목적지 파이프라인 유닛(78)의 각각으로 분리된 메시지를 순차적으로 전송하는데, 상기 메시지 각각에는 각각의 목적지 유닛의 어드레스를 포함하는 헤더가 포함되어 있다. 선택적으로, 상기 파이프라인 유닛(78_1)은 하나의 메시지 내에 각각의 목적지 파이프라인 유닛의 어드레스를 포함하는 데

이터 및 헤더를 포함시킴으로써 상기 목적지 파이프라인 유닛(78₁) 각각으로 상기 데이터를 동시에 전송한다. 상기 데이터를 전송하는 방법은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다음으로, 파이프라인 유닛(78_n)은 데이터를 수신한다. 파이프라인 유닛(78₁~78_n)은 각각 공통의 통신 버스(94)와 결합되기 때문에, 각각의 파이프라인 유닛(78₁~78_n)은 상기 데이터의 의도된 수신처인지 여부를 결정한다. 예를 들어, 각각의 파이프라인 유닛(78₁~78_n)은 자신의 어드레스가 메시지의 헤더에 포함되어 있는지를 결정한다. 이 예에서, 상기 유닛(78₂~78_{n-1})은 이들이 상기 데이터의 의도된 수신처가 아님을 결정하고, 따라서 데이터를 무시한다. 즉, 데이터를 그들의 데이터 메모리(92)(도 4)로 로드하지 않는다. 반대로, 파이프라인 유닛(78_n)은 데이터의 의도된 수신처인지를 결정하여 상기 데이터를 자신의 메모리(92)로 로드한다. 상기 데이터를 수신하는 방법은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

그러면, 파이프라인 유닛(78_n)은 수신된 데이터, 파이프라인 유닛(78₁) 등의 피어, 또는 외부 장치(도시하지 않음)를 처리할 준비를 하고, SYNC 신호를 펄스로 만들어 파이프라인 유닛(78_n)이 적절한 방식으로 상기 데이터를 처리하게 한다. SYNC 신호를 펄스로 만드는 상기 피어/장치는 파이프라인 유닛(78_n)이 수신된 데이터를 처리할 준비가 되었는지를 결정한다. 예를 들어, 상기 피어/장치는, 파이프라인 유닛(78₁)이 데이터를 전송한 후 상기 SYNC 신호를 미리결정된 시간으로 펄스로 만들지만 하던 된다. 상기 미리결정된 시간이 파이프라인 유닛(78_n)이 상기 데이터를 수신하고 그의 데이터 메모리(92)(도 4)에 로드하기에 충분히 길다고 가정한다. 선택적으로, 파이프라인 유닛(78_n)은 SYNC 신호를 펄스로 만들어 상기 피어/장치에게 상기 수신된 데이터를 처리할 준비가 되었음을 알려준다.

다음으로, 펄스로 된 SYNC 신호에 응답하여, 파이프라인 유닛(78_n)이 상기 수신된 데이터를 처리한다. 파이프라인에 의해 상기 데이터를 처리하는 것은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

이어서, 파이프라인 유닛(78_n)이 상기 데이터 처리를 완료하면, 피어, 외부 장치(도시하지 않음) 또는 유닛(78_n) 그 자체가 SYNC 신호를 펄스로 만들어 파이프라인 유닛(78₁)에게 그 이상의 데이터를 전송하도록 통보한다.

두번째 실시예에서, 호스트 프로세서(42)(도 3)는 비-시간-임계 방식으로 데이터를 처리하는 파이프라인(78_n)으로 데이터를 전송한다. 따라서, 호스트 프로세서 및 파이프라인 유닛(78_n)은 하나 또는 그 이상의 이벤트를 사용해서 앞서 설명한 바와 같은 이유로 인해 데이터 전송 및 처리를 동기화 한다.

먼저, 호스트 프로세서(42)(도 3)은 파이프라인 버스(50)(도 3)상에 데이터를 구동시킴으로써 파이프라인 유닛(78_n)으로 데이터를 전송한다. 일반적으로,

호스트 프로세서(42)는 상기 데이터를 포함하는 메시지 및 파이프라인 유닛(78_n)의 어드레스를 포함하는 헤더를 생성한다. 단일 호스트 프로세서(42)가 다중 파이프라인 유닛(78)으로 데이터를 전송하게 되면, 상기 첫번째 실시예에서 설명한 바와 같이 두 가지 방법 중 하나의 방법으로 하게 된다.

다음으로, 파이프라인 유닛(78_n)은 산업-표준 버스 인터페이스(91) 및 통신 버스(94)를 통해 파이프라인 버스(50)(도 3)로부터 데이터를 수신한다. 파이프라인 유닛(78₁~78_n)은 각각 공통의 통신 버스(94)와 결합되어 있으므로, 각각의 파이프라인 유닛은 상기 첫번째 실시예에서 설명한 바와 같은 방식으로 데이터의 의도된 수신처인지를 결정한다.

그리고 나서, 파이프라인 유닛(78_n)은 수신된 데이터를 처리할 준비를 하고, 파이프라인 유닛(78₁) 등의 피어, 또는 외부 장치(도시하지 않음)가 파이프라인 버스(50)상으로 또는 공통 버스(94)상으로 적절 이벤트를 생성하여 파이프라인 유닛(78_n)이 적절한 방식으로 상기 데이터를 처리하게 한다. 상기 이벤트를 생성하는 피어/장치는 파이프라인 유닛(78_n)이 수

신된 데이터를 처리할 준비가 되었는지를 결정한다. 예를 들어, 상기 피어/장치는, 호스트 프로세서(42)가 데이터를 전송한 후 미리 결정된 시간에서 이벤트를 생성하기만 한다. 상기 미리결정된 시간이 파이프라인 유닛(78_n)이 상기 데이터를 수신하고 그의 데이터 메모리(92)(도 4)에 로드하기에 충분히 길다고 가정한다. 선택적으로, 파이프라인 유닛(78_n)은 이벤트를 생성하여 상기 피어/장치에게 상기 수신된 데이터를 처리할 준비가 되었음을 알려준다.

다음으로, 파이프라인 유닛(78_n)이 상기 이벤트를 수신한다. 상기 이벤트를 수신하는 것에 대해서는 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

수신된 이벤트에 응답하여, 파이프라인 유닛(78_n)은 수신된 데이터를 처리한다. 파이프라인 유닛(78_n)에 의해 데이터를 처리하는 것은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

이어서, 파이프라인 유닛(78_n)이 상기 데이터 처리를 완료하면, 피어, 외부 장치(도시하지 않음) 또는 유닛(78_n) 그 자체가 이벤트를 생성하여 호스트 프로세서(42)(도 3)에게 그 이상의 데이터를 전송하도록 통보한다.

도 6을 계속 참고하면, 가속기(44)의 선택적인 수행이 나타나 있다. 예를 들어, 비록 상기 설명한 제1 및 제2 연산 예에서 각각 SYNC 신호와 이벤트를 배타적으로 사용하였으나, 가속기(44)는 SYNC 신호와 이벤트를 조합하여 사용할 수도 있다. 또한, 다른 피어가 하나 또는 그 이상의 다중 파이프라인 유닛(78 또는 100)을 사용하여 그들 각각의 데이터 메모리(92)에 데이터의 벨르 저장할 한다. 또한, 설계자는 호스트 프로세서(42)(도 3)를 하나 또는 그 이상의 파이프라인 유닛(78 또는 100)으로 대체할 수 있는데, 이 유닛은 호스트 프로세서의 기능을 수행하는 "호스트" 피어를 함께 형성한다. 또한, 하나 또는 그 이상의 파이프라인 유닛(78 또는 100)은 하나 또는 그 이상의 메시지-배분 피어로서 작동한다. 예를 들어, 호스트 프로세서(42)가 다중 서브스크라이버 피어로의 전송을 위한 메시지를 생성한다고 가정한다. 호스트 프로세서(42)는 이 메시지를 상기 서브스크라이버 피어 각각으로 메시지를 분배하는 메시지-분배 피어로 전송한다. 따라서, 호스트 프로세서(42)가 아니라 상기 메시지-분배 피어가 상기 메시지 분배를 부담하여 호스트 프로세서가 다른 태스크에 시간과 리소스를 사용하게 한다.

도 7은 본 발명의 다른 실시예에 따른 다중 파이프라인 유닛(78)(도 4) 또는 (100)(도 5)를 갖는 가속기(44)(도 3)의 블록 다이어그램이다.

도 7의 가속기(44)는 파이프라인 유닛(78₁-78_n)과 호스트 프로세서(42)(도 3) 등의 다른 피어들 및 상기 파이프라인 버스(50)(도 3)와 결합된 장치(도시하지 않음) 사이에 데이터를 라우팅하기 위한 통신-버스 라우터(100)가 포함되어 있는 것을 제외하고는 도 6의 가속기(44)와 동일하다. 간략화를 위해, 도 7의 가속기(44)를 다수의 파이프라인 유닛(78₁-78_n)을 갖는 것으로 설명하였으나, 상기 가속기는 다수의 파이프라인 유닛(100) 또는 유닛(78,100)의 조합로 포함할 수 있음을 이해할 수 있다.

상기 통신-버스 라우터(110)는 통신 버스(94)의 각각의 브랜치(94₁-94_n)를 통해 파이프라인 유닛(78₁-78_n)과 결합되어 있고, 버스(112)를 통해 산업-표준 버스 인터페이스(91)(존제한다면)와 결합되어 있다. 선택적으로, 도 6을 참고로 앞서 설명한 바와 같이, 각각의 파이프라인 유닛(78₁-78_n)에는 보드상에 각각의 인터페이스(91)가 포함되어 있고, 그래서 외부 인터페이스(91)를 생략할 수 있어서 라우터(110)가 직접 도 3의 파이프라인 버스(50)(또는 존재한다면 라우터(61))와 결합된다.

라우터(110)는 파이프라인 버스(50)(도 3)로부터 각각의 목적지 파이프라인 유닛 또는 유닛들(78₁-78_n)까지 신호를 라우트하고, 소스 파이프라인 유닛에서부터 하나 또는 그 이상의 목적지 파이프라인 유닛까지 또는 파이프라인 버스까지 신호를 라우트하기도 한다. 따라서, 라우터(110)는 파이프라인 유닛(78₁-78_n)의 각각에서 통신 버스(94)상의 신호가 그 파이프라인 유닛으로 의도되었는지를 결정하는 기능을 오프로드한다. 이 오프로드는 각각의 파이프라인 유닛(78₁-78_n)의 파이프라인 회로(80)(도 4)상의 리소스를 자유롭게 해 주어 각각의 파이프라인 유닛의 기능을 증가시킨다.

도 7을 계속 참고하여, 지금부터 라우터(110)가 있는 다중-파이프라인-유닛 가속기(44)의 동작을 설명한다. 상기 동작은 도 6의 가속기(44)를 위해 앞에서 설명한 바와 유사하기 때문에, 아래 설명은 도 6과 도 7의 가속기간의 동작 차이점을 강조한다.

첫번째 실시예에서, 파이프라인 유닛(78_i)은 시간-일체 방식으로 데이터를 처리하는 파이프라인(78_o)으로 데이터를 전송한다. 그래서, 파이프라인 유닛(78_i-78_j)이 하나 또는 그 이상의 SYNC 신호를 사용하여 도 6의 첫번째 실시예를 통해 설명한 바와 같은 데이터 전송 및 처리를 동기화 한다.

먼저, 파이프라인 유닛(78_i)은 데이터를 통신 버스(94)의 브랜치(94_i)상으로 구동시킴으로써 파이프라인 유닛(78_o)으로 데이터를 전송한다. 일반적으로, 파이프라인 유닛(78_i)은 데이터를 포함하는 메시지 및 상기 파이프라인 유닛(78_o)의 어드레스를 포함하는 헤더를 생성한다.

다음으로, 라우터(110)가 데이터를 수신하고, 상기 데이터의 목적지가 파이프라인 유닛(78_o)인지를 결정하고, 상기 데이터를 통신 버스의 브랜치(94_o)상으로 구동시킨다. 일반적으로, 라우터(110)는 상기 데이터를 포함하는 메시지의 헤더를 분석하고 상기 헤더에서 목적지 어드레스를 추출함으로써 데이터의 목적지를 결정한다. 따라서, 라우터(110)는 데이터의 적절한 목적지를 결정하고, 파이프라인 유닛(78_o)은 상기 데이터의 의도된 수신처인지를 결정하지 않고 상기 라우터로부터 상기 데이터를 단지 받기만 할 수 있다. 선택적으로, 파이프라인 유닛(78_i-78_o)은 상기 데이터의 의도된 수신처 여부를 결정하고, 의도된 수신처가 아니라면, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호, 및 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명된 바와 같이, 예외를 생성한다. 파이프라인 유닛(78_o)은 이 예외를 라우터(110), 산업-표준 버스 인터페이스(91)(존재한다면), 라우터(61)(존재한다면), 및 파이프라인 버스(50)(도 3)을 통해 호스트 프로세서(42)(도 3)으로 전송할 수 있다. 파이프라인 유닛(78_o)은 이 예외를 라우터(110), 산업-표준 버스 인터페이스(91)(존재한다면), 라우터(61)(존재한다면), 및 파이프라인 버스(50)(도 3)을 통해 호스트 프로세서(42)(도 3)으로 전송할 수 있다.

그러면, 파이프라인 유닛(78_o)은 상기 데이터를 버스 브랜치(94_o)으로부터 로드한다. 상기 파이프라인에 의한 데이터 로딩은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다음으로, 파이프라인 유닛(78_o)은 수신된 데이터를 처리할 준비를 하고, 파이프라인 유닛(78_i) 등의 피어, 또는 외부 장치(도시하지 않음)는 SYNC 신호를 펄스로 만들어 파이프라인 유닛(78_o)이 상기 도 6의 첫번째 실시예에서 설명한 바와 같은 적절한 방식으로 상기 데이터를 처리하게 한다.

그러면, 펄스로 된 SYNC 신호에 응답하여, 파이프라인 유닛(78_o)이 도 6의 첫번째 실시예에서 설명한 바와 같은 방법으로 상기 수신된 데이터를 처리한다.

이어서, 파이프라인 유닛(78_o)이 상기 데이터 처리를 완료하면, 피어, 외부 장치(도시하지 않음) 또는 유닛(78_o) 그 자체가 SYNC 신호를 펄스로 만들어 파이프라인 유닛(78_i)에게 그 이상의 데이터를 전송하도록 통보한다.

두번째 실시예에서, 호스트 프로세서(42)(도 3)는 파이프라인(78_o)으로 전송한다. 그래서, 호스트 프로세서 및 파이프라인 유닛(78_o)이 하나 또는 그 이상의 이벤트를 사용하여 도 6을 참고로 위에서 설명한 바와 같은 이유로 인해 상기 데이터 전송 및 처리를 동기화한다.

먼저, 호스트 프로세서(42)(도 3)이 상기 데이터를 버스(50)(도 3)상에서 구동시킴으로써 파이프라인 유닛(78_n)으로 데이터를 전송한다. 일반적으로, 호스트 프로세서(42)는 상기 데이터를 포함하는 메시지 및 파이프라인 유닛(78_n)의 어드레스를 포함하는 헤더를 생성한다. 다음으로, 라우터(110)가 산업-표준 버스 인터페이스(91)(존제한다면) 및 버스(112)를 통해 파이프라인 버스(50)(도 3)으로부터 데이터를 수신한다.

그러면, 라우터(110)는 상기 데이터의 목적지가 파이프라인 유닛(78_n)인지를 결정하고, 상기 데이터를 통신 버스의 브랜치(94_n)상으로 구동시킨다. 일반적으로, 라우터(110)는 도 7의 첫번째 실시예에서 설명한 바와 같이 헤더의 목적지를 결정한다. 따라서, 라우터(110)가 데이터의 적절한 목적지를 결정하기 때문에, 파이프라인 유닛(78_n)은 상기 데이터의 의도된 수신처인지를 결정하지 않고 상기 라우터로부터 상기 데이터를 단지 받기만 할 수 있다. 선택적으로, 파이프라인 유닛(78_n)은 상기 데이터의 의도된 수신처 여부를 결정하고, 의도된 수신처가 아니라면, 앞서 언급한 발명의 명칭이 "IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,102호, 발명의 명칭이 "COMPUTING MACHINE HAVING IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/684,053호, 및 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 설명된 바와 같이, 예외를 생성하고, 이 예외를 도 6의 두번째 실시예에서 설명한 바와 같이 호스트 프로세서(42)(도 3)으로 전송할 수 있다.

다음으로, 파이프라인 유닛(78_n)은 상기 데이터를 버스 브랜치(94_n)으로부터 로드한다. 상기 파이프라인에 의한 데이터 로딩은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다음으로, 파이프라인 유닛(78_n)이 수신된 데이터를 처리할 준비를 하면, 파이프라인 유닛(78_n) 등의 피어, 또는 외부 장치(도시하지 않음)는 파이프라인 버스(50)상에 또는 통신 버스의 상기 브랜치(94₁~94_{n-1})의 하나 상에 이벤트를 생성하여 상기 유닛(78_n)이 도 6의 두번째 실시예에서 설명한 바와 같은 적절한 방식으로 데이터를 처리하게 한다.

다음으로, 라우터(110)가 상기 이벤트를 수신하고, 그것이 파이프라인 유닛(78_n)을 위한 것인지를 결정하고, 상기 이벤트를 상기 버스 브랜치(94_n)상에 구동시킨다.

그러면, 파이프라인 유닛(78_n)은 상기 이벤트를 상기 버스 브랜치(94_n)로부터 로드한다. 상기 파이프라인 유닛(78_n)에 의한 이벤트 로딩은 앞서 언급한 발명의 명칭이 "PIPELINE ACCELERATOR FOR IMPROVED COMPUTING ARCHITECTURE AND RELATED SYSTEM AND METHOD" 인 미국 특허출원 제10/683,929호에 더 설명되어 있다.

다음으로, 수신된 이벤트에 응답하여, 파이프라인 유닛(78_n)은 상기 수신된 데이터를 처리한다.

이어서, 파이프라인 유닛(78_n)이 상기 데이터 처리를 완료하면, 피어, 외부 장치(도시하지 않음) 또는 유닛(78_n) 그 자체가 이벤트를 생성하여 호스트 프로세서(42)(도 3)가 데이터 전송을 더 하도록 통보한다.

도 7을 계속 참고하면, 비록 상기 설명한 제1 및 제2 연산 예에서 각각 SYNC 신호와 이벤트를 배타적으로 사용하였으나, 가속기(44)는 SYNC 신호와 이벤트를 조합하여 사용할 수도 있다.

도 8은 본 발명의 일 실시예에 따른 다중 파이프라인 유닛(78)(도 4) 또는 (100)(도 5)의 다수의 그룹(120)을 포함하는 도 3의 가속기(44)의 블록 다이어그램이다. 파이프라인 유닛의 다수 그룹(120)을 포함시키는 것은 가속기(44)의 기능을 증가시켜 설계자로 하여금 관련된 연산을 수행하는 파이프라인 유닛을 그룹평행으로써 가속기의 효율을 증가시키게 한다. 간략화를 위해서, 도 8의 가속기(44)를 여러 파이프라인 유닛(78)을 가지는 것으로 설명하였으나, 상기 가속기에는 다중 파이프라인 유닛(100) 또는 유닛(78)과 유닛(100)의 결합이 포함될 수 있음을 이해할 수 있다. 또한, 파이프라인 유닛(78)에는 비록 다른 실시예에서는 포함되었으나, 산업-표준 버스 인터페이스(91)(이 인터페이스는 본 실시예에서는 외부에 있음)는 포함되지 않는다.

가속기(44)에는 파이프라인 유닛(78)의 6개의 그룹($120_1 \sim 120_6$)이 포함되어 있는데, 각 그룹은 3개의 파이프라인 유닛 및 상기 파이프라인 유닛을 서로 연결하고 다른 파이프라인 유닛 그룹과 연결시키는 각각의 그룹간 통신-버스 라우터($110_1 \sim 110_6$)를 갖는다. 비록 가속기(44)를 3개의 파이프라인 유닛(78) 각각의 6개 그룹($120_1 \sim 120_6$)을 포함하는 것으로 설명하였으나, 가속기의 다른 수행에서는 가상적으로 다른 수의 파이프라인 유닛의 다른 수의 그룹이 포함될 수 있으며, 이들 그룹 모두는 동일한 개수의 파이프라인 유닛을 가질 필요가 있다. 또한, 통신-버스 라우터($110_1 \sim 110_6$)는 도 6의 가속기(44)를 참고하여 앞에서 설명한 바와 같이 생략할 수 있다.

파이프라인-유닛 그룹(120_1)에는 도 7을 참고하여 위에서 설명한 바와 유사한 방식으로 통신 버스의 브랜치($94_1, 94_2, 94_3$) 각각을 통해 그룹간 통신-버스 라우터(110_1)와 연결된 세 개의 파이프라인 유닛($78_1 \sim 78_3$)이 포함되어 있다. 다른 그룹($120_2 \sim 120_6$)도 유사하다.

상기 그룹($120_1 \sim 120_3$)의 통신-버스 라우터($110_1 \sim 110_3$)는 제1-레벨 버스(126_1)의 각각의 브랜치($124_1 \sim 124_3$)를 통해 제1-레벨 라우터(122_1)와 연결되어 있다. 상기 라우터(122_1)와 버스(126_1)는 파이프라인 유닛($78_1 \sim 78_3$)이 서로 통신하게 한다.

유사하게, 상기 통신-버스 라우터($110_4 \sim 110_6$)는 제1-레벨 버스(126_2)의 각각의 브랜치($128_1 \sim 128_3$)를 통해 제1-레벨 라우터(122_2)와 연결된다. 상기 라우터(122_2)와 버스(126_2)는 파이프라인 유닛($78_{10} \sim 78_{18}$)이 서로 통신하게 한다.

상기 제1-레벨 라우터($122_1, 122_2$)는 제2-레벨 버스(134)의 각각의 브랜치($132_1 \sim 132_2$)를 통해 제2-레벨 라우터(130)와 연결된다. 상기 라우터(130)와 버스(134)는 파이프라인 유닛($78_1 \sim 78_{18}$)이 서로 통신하게 하며 후술하는 바와 같이 다른 피어/장치들과 통신하게 한다.

파이프라인 버스(50) 및 2차 파이프라인 버스(136)는 상기 각각의 산업-표준 버스 인터페이스($91_1, 91_2$)를 통해 상기 제2-레벨 라우터(130)와 연결된다. 상기 2차 파이프라인 버스(136)는 호스트 프로세서(42)(도 3)와 같은 피어, 또는 상기 파이프라인 버스(50)와 연결되지 않는 하드-디스크 드라이브(도시하지 않음)와 같은 주변장치들과 연결된다. 또한, 상기 버스(50, 136) 중 어느 하나 또는 둘 모두는, 상기 가속기(44)가 상기 호스트 프로세서(42)(도 3)와 같은 다른 피어로부터 원격 위치할 수 있는 네트워크나 인터넷(도시하지 않음)을 통해 피어나 주변장치들과 결합된다.

버스(138)는 하나 또는 그 이상의 SYNC 신호를 상기 파이프라인 유닛($78_1 \sim 78_{18}$) 모두 및 상기 호스트 프로세서(42)(도 3) 또는 장치(도시하지 않음)와 같은 다른 피어와 직접 접속된다.

도 8을 계속 참고하면, 동작의 한 예에서, 파이프라인 유닛(78)의 각각의 그룹($120_1 \sim 120_6$)은 상기 2차 파이프라인 버스(136)와 결합된 소나 어레이(도시하지 않음)의 센서 각각으로부터 데이터를 처리한다. 상기 그룹(120_1)의 파이프라인 유닛($78_1 \sim 78_3$)이 하나의 라우터(110_1)에 의해 상호 접속되기 때문에, 이들 파이프라인 유닛들은 다른 그룹($120_2 \sim 120_6$)의 파이프라인 유닛($78_4 \sim 78_{18}$)과 통신하는 것 보다 훨씬 빠르게 서로 통신을 한다. 이러한 높은 통신 속도는 또한 다른 그룹($120_2 \sim 120_6$) 각각에서도 존재한다. 따라서, 설계자는 데이터를 자주 전송하거나 그렇지 않으면 그룹간의 통신을 하는 파이프라인 유닛들을 서로 그룹화 함으로써 가속기(44)의 처리 속도를 증가시킬 수 있다.

일반적으로, 파이프라인 유닛($78_1 \sim 78_{18}$)은, 도 7을 참고하여 위에서 설명한 것과 유사한 방식으로, 서로간에 통신을 하고, 상기 호스트 프로세서(42)(도 3)와 같은 피어와 통신을 하며, 버스(50, 136)와 결합된 장치들과 통신을 한다. 예를 들어, 버스(136)에 결합된 센서(도시하지 않음)는 산업-표준 버스 인터페이스(91_1), 제2 레벨 라우터(130), 제1 레벨 라우터(122_1), 및 그룹간 라우터(110_1)를 통해 파이프라인 유닛(78_1)과 통신을 한다. 비슷하게, 파이프라인 유닛(78_1)은 라우터($110_1, 122_1, 110_2$)를 통해 파이프라인 유닛(78_2)과 통신을 하고, 라우터($110_1, 122_1, 130, 122_2, 110_3$)를 통해 파이프라인 유닛(78_{10})과 통신을 한다.

앞의 설명으로부터 당업자는 본 발명을 실시하거나 활용할 수 있다. 본 발명의 실시예들을 당업자는 다양하게 변형시킬 수 있다는 것은 분명하고, 본 발명의 요지와 범위를 벗어나지 않고 다른 실시예 및 응용 분야에 적용할 수 있다. 따라서, 본 발명의 실시예들은 본 발명을 제한하기 위한 것이 아니고 여기에서 공개한 원리와 특징에 넓게 해석되어야 할 것이다.

(57) 청구의 범위

청구항 1.

통신 버스; 및

상기 통신 버스와 각각 결합되어 있고, 각각 하드웨어드-파이프라인 유닛을 포함하는 다수의 파이프라인 유닛을 구비하는 것을 특징으로 하는 파이프라인 가속기.

청구항 2.

청구항 1에 있어서,

상기 파이프라인 유닛 각각은,

상기 하드웨어드-파이프라인 회로와 결합되어 있는 각각의 메모리를 포함하며,

상기 하드웨어드-파이프라인 회로는,

상기 통신 버스로부터 데이터를 수신하고,

상기 데이터를 상기 메모리로 로드하고,

상기 메모리로부터 상기 데이터를 검색하고,

상기 검색된 데이터를 처리하고,

상기 처리된 데이터를 상기 통신 버스상에서 구동시키도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

청구항 3.

청구항 1에 있어서,

상기 파이프라인 유닛 각각은,

상기 하드웨어드-파이프라인 회로와 결합되어 있는 각각의 메모리를 포함하며,

상기 하드웨어드-파이프라인 회로는,

상기 통신 버스로부터 데이터를 수신하고,

상기 데이터를 처리하고,

상기 처리된 데이터를 상기 메모리로 로드하고,

상기 메모리로부터 상기 처리된 데이터를 검색하고,

상기 검색된 데이터를 상기 통신 버스상에 로드하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

청구항 4.

청구항 1에 있어서,

상기 하드와이어드-파이프라인 회로는 각각의 펄드-프로그램가능한 게이트 어레이상에 배치되는 것을 특징으로 하는 파이프라인 가속기.

청구항 5.

청구항 1에 있어서,

파이프라인 버스; 및

상기 통신 버스 및 상기 파이프라인 버스와 결합되어 있는 파이프라인-버스 인터페이스를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

청구항 6.

청구항 1에 있어서,

상기 통신 버스는 각각 개별적인 파이프라인 유닛과 결합된 다수의 브랜치를 포함하는 것을 특징으로 하는 파이프라인 가속기.

청구항 7.

청구항 1에 있어서,

상기 통신 버스는 각각의 파이프라인 유닛과 결합된 다수의 브랜치를 포함하며,

상기 브랜치 각각과 결합된 라우터;

파이프라인 버스; 및

상기 라우터 및 상기 파이프라인 버스와 결합된 파이프라인-버스 인터페이스를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

청구항 8.

청구항 1에 있어서,

상기 통신 버스는 각각의 파이프라인 유닛과 결합된 다수의 브랜치를 포함하며,

상기 브랜치 각각과 결합된 라우터;

파이프라인 버스;

상기 라우터 및 상기 파이프라인 버스와 결합된 파이프라인-버스 인터페이스; 및

상기 라우터와 결합된 2차 버스를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

청구항 9.

청구항 1에 있어서,

상기 통신 버스는 상기 파이프라인 유닛의 하나로 어드레스된 데이터를 수신하도록 동작 가능하며,

상기 하나의 파이프라인 회로는 상기 데이터를 받도록 동작 가능하며,

상기 다른 파이프라인 회로는 상기 데이터를 거절하도록 동작 가능한 것을 특징으로 하는 파이프라인 가속기.

청구항 10.

청구항 1에 있어서,

상기 통신 버스는 각각의 파이프라인 유닛과 결합된 다수의 브랜치를 포함하며,

상기 브랜치 각각과 결합되어 있으며,

상기 파이프라인 유닛의 하나로 어드레스된 데이터를 수신하고,

상기 통신 버스의 상기 각각의 브랜치를 통해 상기 하나의 파이프라인 유닛으로 상기 데이터를 제공하도록 동작 가능한 라우터를 더 구비하는 것을 특징으로 하는 파이프라인 가속기.

청구항 11.

프로세서;

상기 프로세서와 결합된 파이프라인 버스; 및

파이프라인 가속기를 구비하며,

상기 파이프라인 가속기는,

통신 버스,

상기 파이프라인 버스와 상기 통신 버스 사이에 결합된 파이프라인-버스 인터페이스, 및

상기 통신 버스와 각각 결합되어 있으며 각각 하드웨어로-파이프라인 회로를 포함하는 다수의 파이프라인 유닛을 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 12.

청구항 11에 있어서,

상기 프로세서는 상기 파이프라인 유닛의 하나를 식별하는 메시지를 생성하고 상기 메시지를 상기 파이프라인 버스상에서 구동시키도록 동작 가능하고,

상기 파이프라인-버스 인터페이스는 상기 메시지를 상기 통신 버스와 결합하도록 동작 가능하고,

상기 파이프라인 유닛은 각각 상기 메시지를 분석하도록 동작 가능하고,

상기 식별된 파이프라인 유닛은 상기 메시지를 받도록 동작 가능하며,

상기 다른 파이프라인 유닛은 상기 메시지를 거절하도록 동작 가능한 것을 특징으로 하는 컴퓨팅 머신.

청구항 13.

청구항 11에 있어서,

상기 통신 버스는 각각의 파이프라인 유닛과 결합된 다수의 브랜치를 포함하며,

상기 프로세서는 상기 파이프라인 유닛의 하나를 식별하는 메시지를 생성하고 이 메시지를 상기 파이프라인 버스상에서 구동시키도록 동작 가능하며,

상기 브랜치 각각 및 상기 파이프라인-버스 인터페이스와 결합되어 있으며 상기 파이프라인-버스 인터페이스로부터 상기 메시지를 수신하고 상기 식별된 파이프라인 유닛으로 상기 메시지를 제공하도록 동작 가능한 라우터를 더 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 14.

청구항 11에 있어서,

상기 통신 버스는 각각의 파이프라인 유닛과 결합된 다수의 브랜치를 포함하며,

2차 버스, 및

상기 브랜치 각각, 상기 파이프라인-버스 인터페이스, 및 상기 2차 버스와 결합된 라우터를 더 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 15.

통신 버스를 통해, 각각의 하드웨어드 파이프라인을 포함하는 다수의 파이프라인 유닛의 제1 파이프라인 유닛으로 데이터를 전송하는 단계; 및

상기 제1 파이프라인 유닛을 가지고 상기 데이터를 처리하는 단계를 구비하는 것을 특징으로 하는 방법.

청구항 16.

청구항 15에 있어서,

상기 데이터를 전송하는 단계는,

상기 데이터를 라우터로 전송하는 단계, 및

상기 통신 버스의 각각의 제1 브랜치를 통해 상기 라우터를 가지고 상기 제1 파이프라인 유닛으로 상기 데이터를 제공하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 17.

청구항 15에 있어서,

상기 데이터를 전송하는 단계는,

프로세서를 가지고 상기 제1 파이프라인 유닛으로 상기 데이터를 전송하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 18.

청구항 15에 있어서,

상기 데이터를 전송하는 단계는,

상기 다수의 파이프라인 유닛의 제2 파이프라인 유닛을 가지고 상기 제1 파이프라인 유닛으로 상기 데이터를 전송하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 19.

청구항 15에 있어서,

상기 제1 파이프라인 유닛을 가지고 상기 처리된 데이터를 상기 통신 버스상에서 구동시키는 단계를 더 구비하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 20.

청구항 15에 있어서,

상기 제1 파이프라인 유닛을 가지고 상기 데이터를 처리하는 단계는,

상기 하드웨어-파이프라인 회로를 가지고 상기 통신 버스로부터 상기 데이터를 수신하는 단계,

상기 하드웨어-파이프라인 회로를 가지고 상기 데이터를 메모리에 로드하는 단계,

상기 하드웨어-파이프라인 회로를 가지고 상기 메모리로부터 상기 데이터를 검색하는 단계, 및

상기 하드웨어-파이프라인 회로를 가지고 상기 검색된 데이터를 처리하는 단계를 포함하는 것을 특징으로 하는 컴퓨팅 머신.

청구항 21.

청구항 15에 있어서,

상기 제1 파이프라인 유닛을 가지고 상기 데이터를 처리하는 단계는,

하드와이어드-파이프라인 회로를 가지고 상기 통신 버스로부터 상기 데이터를 수신하는 단계,

상기 하드와이어드-파이프라인 회로를 가지고 상기 수신된 데이터를 처리하는 단계,

상기 하드와이어드-파이프라인 회로를 가지고 상기 처리된 데이터를 메모리로 로드하는 단계, 및

상기 메모리로부터 상기 처리된 데이터를 검색하고, 상기 하드와이어드-파이프라인 회로를 가지고 상기 처리된 데이터를 상기 통신 버스상에서 구동시키는 단계를 포함하는 것을 특징으로 하는 컴퓨터 시스템.

청구항 22.

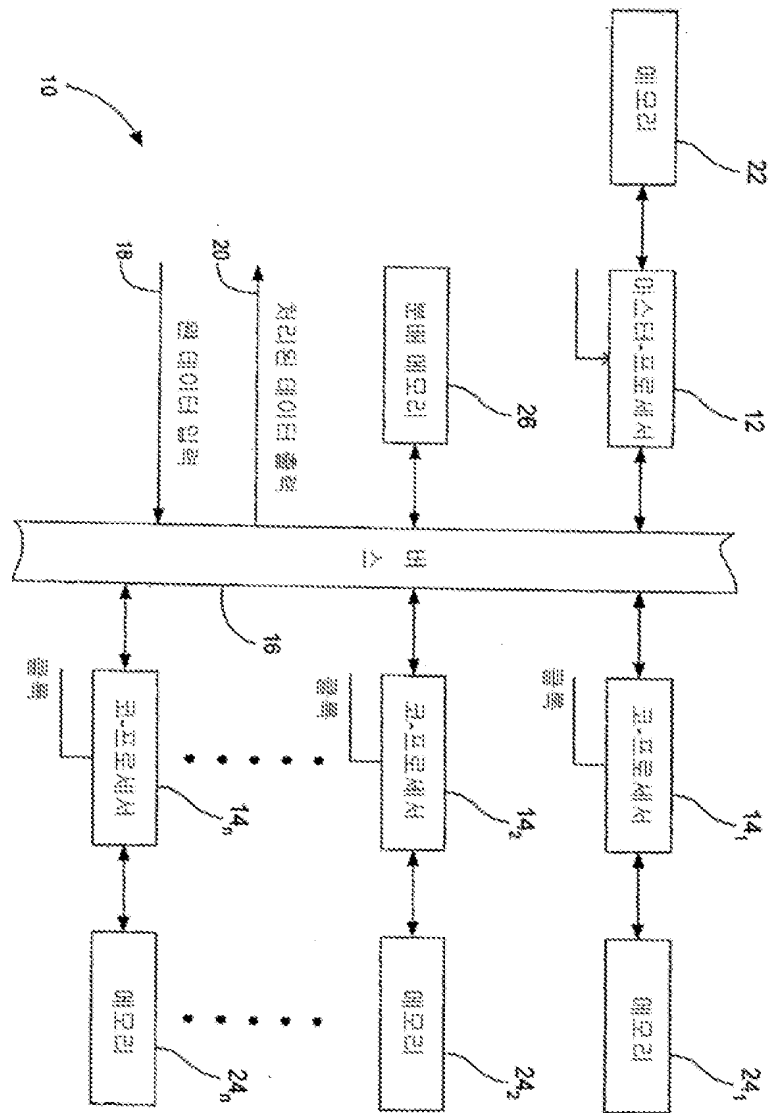
청구항 15에 있어서,

상기 데이터를 포함하고, 상기 메시지의 수신자로서 상기 제1 파이프라인 유닛을 식별하는 메시지를 생성하는 단계를 더 구비하고,

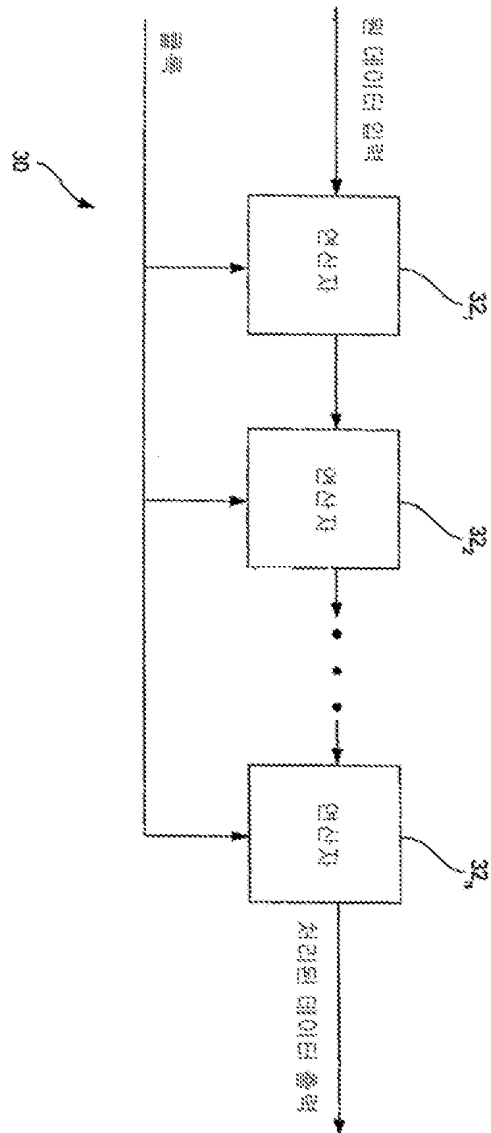
상기 데이터를 상기 제1 파이프라인 유닛으로 전송하는 단계는, 상기 제1 파이프라인이 상기 메시지의 수신자인 메시지를 통해 결정하는 단계를 포함하는 것을 특징으로 하는 컴퓨터 시스템.

도면

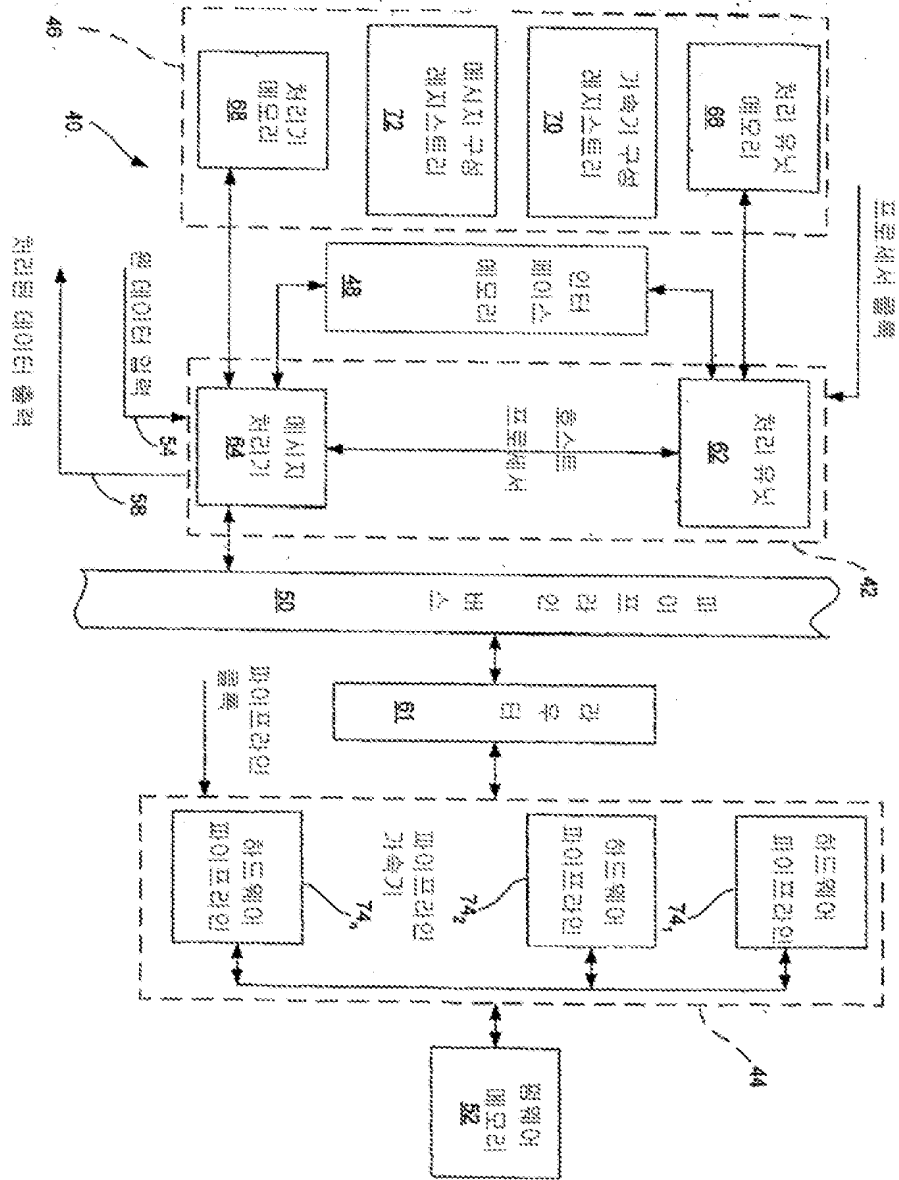
도면1



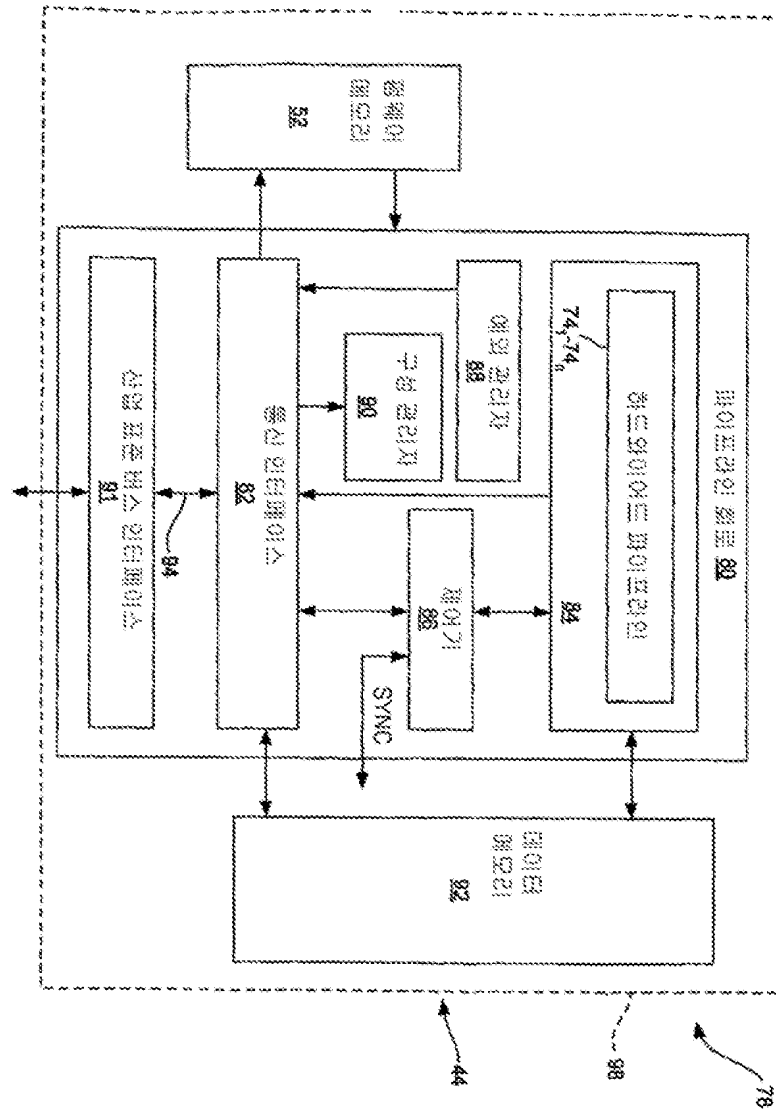
도면2



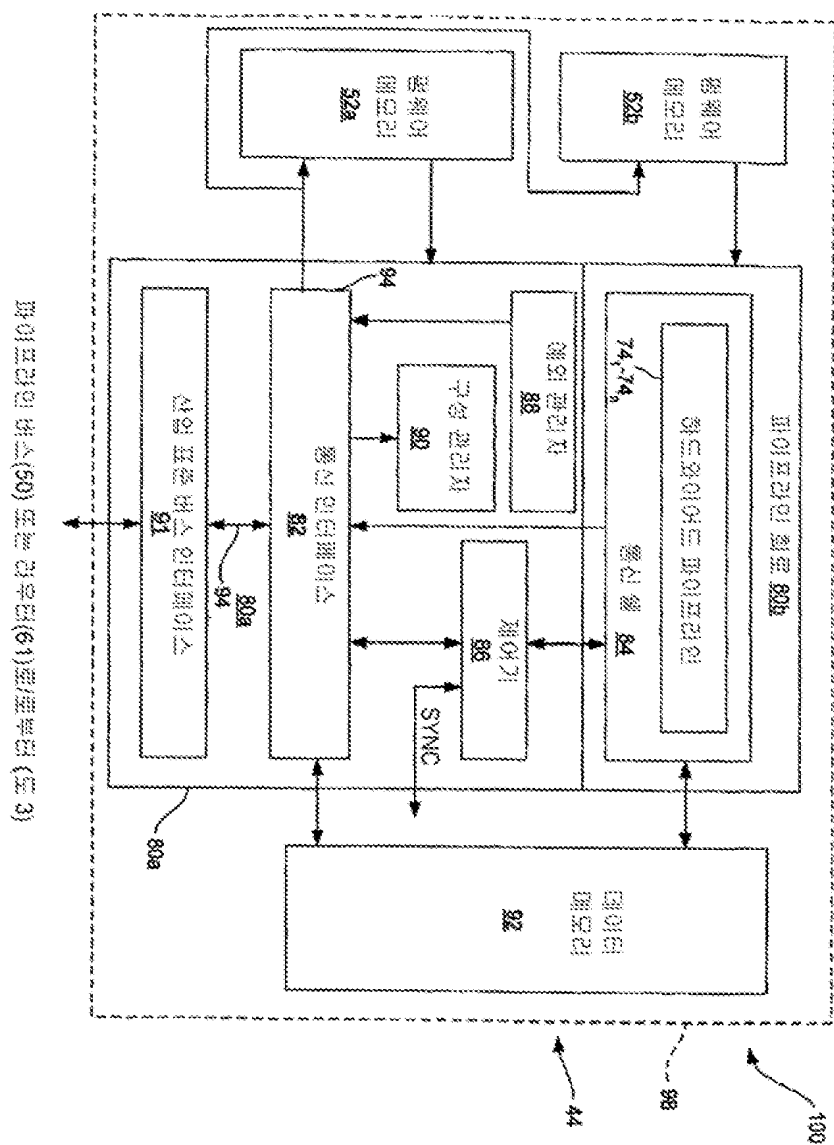
도면3



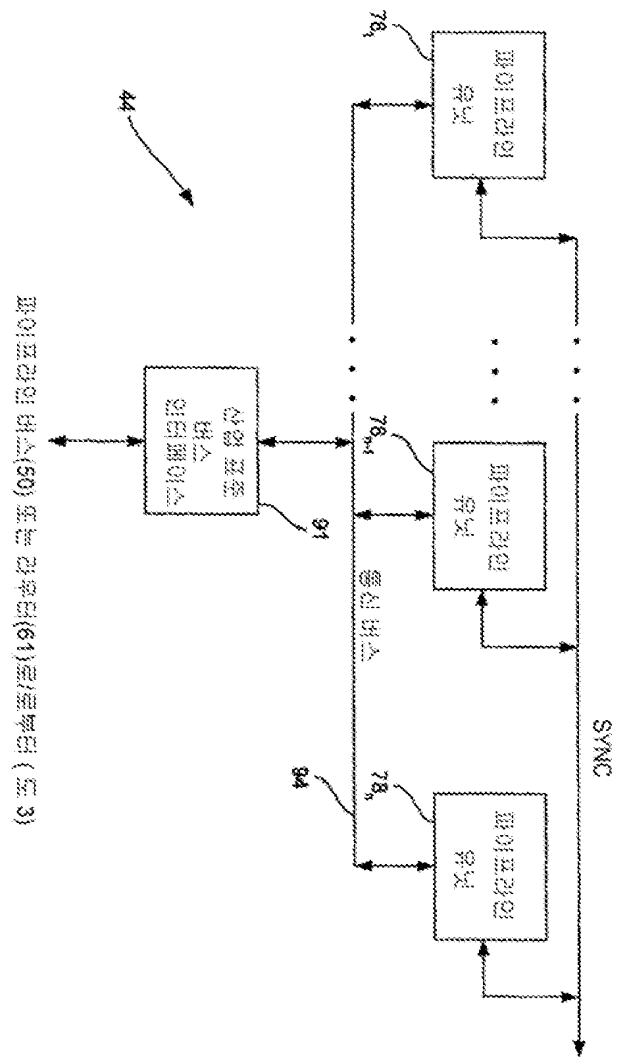
도면4



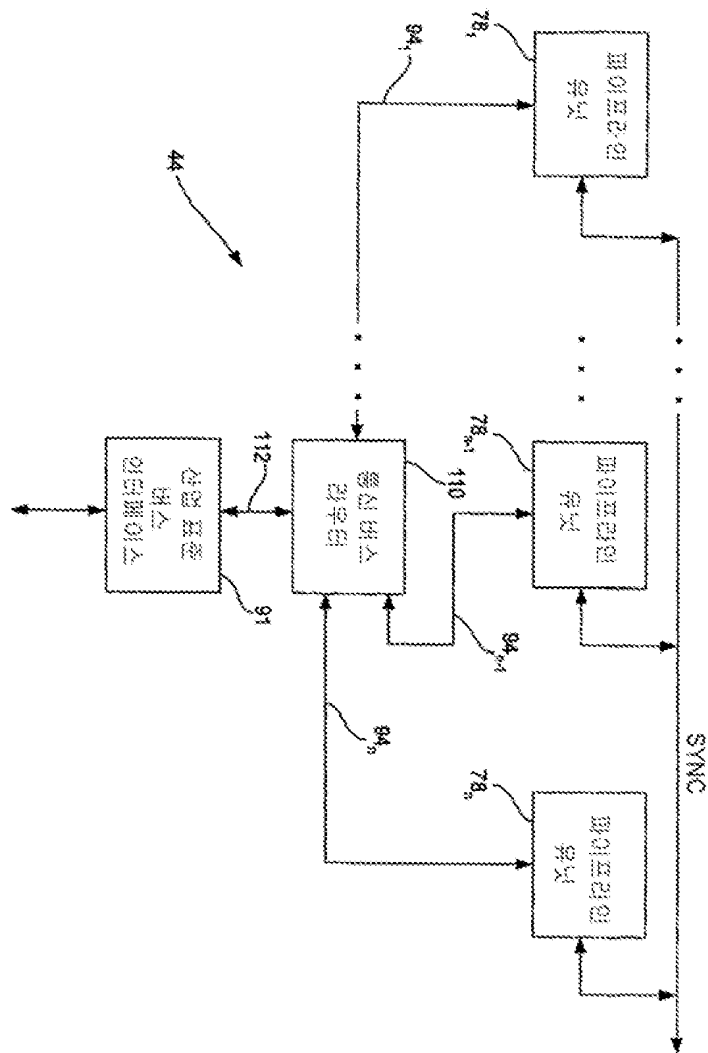
596



도면6



도면7



도면8

